Distributed monitoring of heterogeneous robotic cells. A proposal for the Footwear Industry 4.0.

ARTICLE HISTORY
Compiled August 12, 2018

ABSTRACT
In the footwear sector, automation is often performed by robotic arms due to the inherent adaptive requirements of some of its tasks. With the Industry 4.0 revolution, automation is even less human dependent than before. The fact that fewer people interact with machinery during the manufacturing process has led to a rising need for production monitoring. The interconnection of machines present in 4.0 environments makes monitoring possible at a low investment cost. In this paper, a common communication layer is proposed to enable homogeneous status data retrieval of robotic arms from several different robot manufacturers. This layer is then attached to a 3D simulator software as a client, responsible for the monitoring process. Experiments proved the feasibility of the proposed method in a test bench scenario applying the required constraints of time, cost, bandwidth and disk space.

KEYWORDS
Monitoring; Simulation; Robotics; Footwear; Industry 4.0.

1. Introduction

The footwear industry has, to a certain extent, taken advantage of all industrial revolutions. The first revolution took hold during the eighteenth and nineteenth century. The industrial revolution led to the reduction of work time necessary to transform a natural resource into a useful product. Industrial machinery then began to be powered by the steam engine which also facilitated the transportation of materials and finished products (Wahlster, 2012).

The Second Industrial Revolution began in the first decade of the twentieth century. This time, brought about the electrification of factories and the use of fossil fuels. A reduction in costs and an increase in production and profits was achieved in the manufacturing process (Segal & Thomson, 1990).

The Third Industrial Revolution started in the decade of the 1970's. With the incorporation of electronics and IT, the aim was to achieve further automation in manufacturing. The firsts programmable logic controllers (PLC) and robotic arms boosted the autonomy of industrial tasks. Dynamic processes took advantage of this revolution thanks to the adaptive nature of the technologies introduced.

This revolution was especially significant in industries such as footwear, where a high degree of task flexibility is required because of the diversity of shoe models, different sizes of the same model with left and right feet, the quantity of different parts that make up the shoe and the large number of operations involved in transforming the materials into a finished product (Maurtua, Goenaga, & Tellaeche, 2012).

Another important factor is the high degree of precision needed in some of the tasks

to meet industry requirements (Davia, Jimeno-Morenilla, & Salas, 2013).

A number of footwear tasks have been studied and automated thanks to the third revolution. Examples of work done in footwear automated tasks include the application of adhesive to shoe soles (Wu, 2008), roughing or grinding of the bottom surface of the shoe so that the gluing process works correctly (Pedrocchi, Villagrossi, Cenati, & Tosatti, 2015; Jatta, Zanoni, Fassi, & Negri, 2004), last milling to create shoe lasts based on custom measurements (Xiong, Zhao, Jiang, & Dong, 2010), last marking to add information to them (Conte, Facchino, Marziali, & Mazzocchetti, 2013), lasting operation to secure the leather upper to the sole (Nemec, Lenart, & Zlajpah, 2003), polishing of the finished product (Zlajpah & Nemec, 2008) and shoe-packaging (Gracia et al., 2017; Perez-Vidal et al., 2018) at the end of the production line. Those processes require a high degree of flexibility to face continuous changes in product demand and variety, and to manage complex and variant production processes (Brusaferri, Ballarino, & Carpanzano, 2011).

Recently, a new revolution called Industry 4.0 has surfaced to address current challenges of shorter product life-cycles, highly customised products and stiff global competition. The new manufacturing paradigm consists in modular factory structures composed of smart devices within a networked Internet of Things (IoT) environment. Currently rigid planning and production processes can now be revolutionized (Kargermann & Achatz, 2010).

As several tasks run at the same time within automated 4.0 footwear factories, it can be necessary to monitor all of them to ensure correct production flow and to fix possible unexpected problems during the workday.

Monitoring systems designed for footwear environments should come at a low investment cost, because footwear manufacturing is largely comprised of small and medium sized enterprises. However, current monitoring systems can be expensive or have limited functionalities that do not scale with adaptive shoe factories (Carpanzano & Jovane, 2007). This is true for other industrial sectors, but it is a must in the footwear industry. There exists a huge lack of automation in shoe factories, and the low resources available to invest in increase automation will be prioritized and spent on other technologies such as robotic cells and specific machinery to enhance production, leaving other topics such as monitoring or even safety in background. (Hsing, 1999)

In addition, thanks to the implicit decentralisation inherent to 4.0 environments which enables the different systems inside the smart factory to be autonomous, the monitoring system can be used to monitor all the various processes even if they are spread around the world (Jazdi, 2014).

The main objective of this study was to design, build and test a low-cost monitoring system based on software. It had to focus specifically on Industry 4.0 footwear factories based on robotic arm cells. Scalability, low-cost, safety and security were key factors in the design of the proposed method. Low-cost comes from reusing robotic simulation and cell design software to be an enhanced 3D monitoring client, using the wired and/or wireless network already existent in Industry 4.0 environments and avoiding expensive and complex video camera installations on the factories. This method is especially useful in robotised dynamic tasks, where the process depends on the input of the cell and not on a set of offline programmed paths, which can lead to undesired safety problems because of uncontrolled robot movements due to input noise or other software related problems. However, it can also be used to monitor other rigid robotic cells of the factories based on offline paths. Examples of both kind of robotic cells can be found in the experiment section, under the prototype cell description subsection. Although the system has been designed taking into account the particularities of the footwear sector,

technology transfer can be made to other sectors with similar properties such as, but not limited to, food (Bogue, 2009), toy (Vincze & Pichler, 2002), aerospace (Summers, 2005) or furniture (Larsson & Kjellander, 2006). Due to the lack of competence of footwear SMEs concerning matters of Industry 4.0, the experiments in this paper have been performed in a laboratory environment, to test the proposed method before it can be implemented in real life scenarios (Uhlemann, Schock, Lehmann, Freiberger, & Steinhilper, 2017).

This article is organized as follows: section 2 gives a short state-of-the-art review on robotic cell monitoring for footwear applications. The need for monitoring in these Industry 4.0 environments is also explained. Section 3, describes how the connection between robotic arms from different manufacturers are unified. Section 4 shows how a simulator software can be used to monitor those robots. In section 5, the created prototype is described as well as the tests performed, and conclusions are drawn in section 6.

2. Monitoring in advanced factories

Being a subject undergoing intense study, monitoring and technologies that allow it have been reviewed by other authors of different fields.

One of the core parts of a monitoring system is the ability to connect and fetch real-time data from the machines to be monitored. Other authors rely on eXtensive Markup Language, commonly known as XML, to encode the transmitted data from monitored machinery (Mantha, Menassa, & Kamat, 2018). This standard works well in cases where data transmission is carried out sporadically and when messages need meta-data (Michalos et al., 2015). The verbosity of XML becomes a problem when data needs to be fetched at high rate and in real-time and also if this data needs to be stored for later use. This paper discusses a custom-made raw protocol that simplifies the fetching of data and makes this process use less bandwidth and storage space.

Once the information is fetch out from the physical devices there has to be modules to manage and visualize it. Simulation in manufacturing environments allows a wide range of solutions: product design, functional analysis and optimisation, floor control, etc. Monitoring systems based on simulation for low rate data fetch based on MES (Manufacturing Execution System) (Mourtzis, Papakostas, Mavrikios, Makris, & Alexopoulos, 2015) have already been studied, but these systems fail to deliver high rate real-time data to reproduce the motion of robotic automated cells. Simulation can also be used to design and test robotic cells environments both for static and mobile robotics solutions (Michalos et al., 2014).

Simulation is used by other authors to perform Virtual Commissioning (Makris, Michalos, & Chryssolouris, 2012), that allows to create a virtual clone of a physical machine to test how will it perform in production before buying or installing it. However, these systems do not monitor real physical devices, but they supplant them for analysis.

Simulation can also be a useful tool for monitoring purpose as it enhances the way users can access, manage and visualise data. This has been used previously as part of Digital-Twin approaches where physical devices and virtual clones are synchronised to have the same information (Uhlemann, Lehmann, & Steinhilper, 2017). However, those systems have a high degree of complexity and its implementation tends to be expensive, which goes against the reality of SMEs that power the footwear sector. This is one of the reasons of making the proposed system, to allow the implementation in

factories of a low-cost tool focused in monitoring that can be expanded according the user needs, leaving out other digital twin aspects that can increase the cost of the product. Also, implementation details shown in the current paper aim to ease the reproducibility and implementation in factories of the proposed method.

3. Monitoring in footwear 4.0

Monitoring systems applied to industrial environments present the following advantages, among others:

- Ability to replay processes and problems in production
- Reduction of plant operator and maintenance personnel
- Remote monitoring
- Low cost implementation
- Error and warning real-time notifications
- Analysis of gathered data

The generic list of advantages is not footwear or 4.0 exclusive, but the characteristics of both scenarios have been taken into account in the design process of the proposed method as it is explained in this section.

The ability to retrieve the status data from the robotic arm during the monitoring process allows the user to save it for later use. This data can be used to replay the process and analyse it offline. It also allows the user to share the replay with other users to help in the analysis process or to report the failure. In case of error, the source of the problem can be located combining the visual inspection of the motion simulated using the replay data and the notifications received. This can be used to address some of the problems that can arise in the cell and avoid random guesses (Gungor & Hancke, 2009). The proposed system cannot monitor all the possible errors of the robotic cell, but the user might add the most important or frequent ones to the monitoring system so it can handle them. The proposed method is focused on footwear factories where the main automation devices are robotic arms. If the error is raised by other hardware element different than the robot, a condition can be coded in the robotic cell controller, so it performs a stop of the robot or throw a signal that can be routed to the monitoring station.

The existence of a centralised monitoring station can lead to staff reduction. Instead of several workers physically checking all of the automated work stations, a minimal group of them could monitor the entire factory. Shoe factories can benefit from this as they tend to have a large number of workers who could be producing more valuable work on other, more productive tasks instead of having to monitor the automated machinery. Another option is to reduce the worker count inside the factory, as less are needed if the monitoring can be performed remotely from anywhere. The impact of this system can escalate better on bigger factories, with a high worker count. Also, the system keeps personnel better informed of the status of the process, reducing downtime and improving safety. (Leo H. Chiang, MS -Richard D. Braatz, 2005)

To avoid having users looking 24/7 at the monitoring screen, the system sends notifications on malfunctions or problems needing attention. When a notification is received, the worker can open the monitor software to connect to the offending cell and perform a check (Gorecky, Schmitt, Loskyll, & Zuhlke, 2014).

Furthermore, the entire monitoring process can be performed remotely from outside the factory through internet services making it possible to know a factory's status from

any location in the world. This is useful in decentralised production scenarios such as that of footwear industry, where each part of the shoe might be crafted in a different factory. This decentralisation is also a feature inherited from I4.0 (Gilchrist, 2016).

The whole monitoring environment can be implemented in the factory at a low investment cost. This is a key requirement in the footwear industry, where a major part of production is ensured by small and medium sized enterprises. There are three reasons this system is low-cost. The first is that the whole monitoring system is based on a 3D simulation software client developed by the authors (Román-Ibáñez, Pujol-López, Mora-Mora, Pertegal-Felices, & Jimeno-Morenilla, 2018), with no extra hardware involved. The second one is that in the shoe factory scenario described herein, the simulation software used as a monitoring client can also be used for cell design and collision testing in robotic arm cells (Zlajpah, 2008). The third one is that having a 4.0 factory means that all robotic arm cells are already connected to the same corporate intranet and this network can be reused for the proposed system. This allows the monitoring client to retrieve the required data from the current 4.0 network, removing the need of interconnect all the cells to monitor them (Perez, Irisarri, Orive, Marcos, & Estevez, 2015). Also, the monitoring system works with the standard TCP/IP and is compatible with wired networks and also with emerging communication technologies, such as 5G networks, that are being applied to the I4.0 (Wollschlaeger, Sauter, & Jasperneite, 2017). Machines working with other technologies such as PROFIBUS can be used if translated to TCP with hardware converters.

Additionally, decision making present in I4.0 can be fed with the information processed using the data gathered in the monitoring sessions as a training set (Fawaz, Merzouki, & Ould-Bouamama, 2009).

The proposed system aims to monitor the status of the automatised robotic cells inside footwear factories, and handle notifications of events produced by them. The monitor client maintains a list of the robotic cells to be monitored, and receive and display notifications, allowing to open a 3D visor to monitor specific cells in detail. The monitoring visor is based on a software simulation software created by the authors, that mimics the motion of the robotic arms inside the monitored cells of the factory by fetching the value of the angles for each joint of the robotic arm. It uses a custom-made communication protocol to fetch data from different robotic arms manufacturers and models, and a simple networking protocol to manage the queries and resulting data. The proposed system can be accessed both from inside the factory and remote locations, considering cybersecurity measures.

4. Homogeneous cell protocol

4.1. *Robotic cell communication*

If a factory is designed from the ground up, it is possible to choose all the factory's robotic arms from the same manufacturer. This decision can make it simpler to deal with the communication aspect of the monitoring process. However, a realistic scenario is that there will be different models of robots and from different manufacturers (Schlechtendahl, Keinert, Kretschmer, Lechler, & Verl, 2015). The factory may already own older robots, upgrade some of them, or add different ones based on task requirements or for purposes of economy.

Users can connect to a robotic arm controller in order to retrieve status information (joint angles, tool position and orientation, error status, etc.) or to send commands

to the arm (move arm to a desired position and orientation, change speed, open attached tool clamp, etc.). Ways of connecting may vary according to the manufacturer. There can even be protocol differences between different models coming from the same manufacturer. These different connection modes can be categorized as:

- Sockets
- SDK/API
- Scripts

Some manufacturers such as Universal Robots use TCP sockets to communicate with the robotic arm. They have several TCP ports open and waiting for commands to be processed. Each data port has a different purpose, for example port 30002 is used to retrieve commands at 10 Hz, while port 30004 is the Real-Time Data Exchange service (RTDE) (*Universal Robots RTDE*, n.d.), which is intended for real-time operations and does the same at 125 Hz. It is important to note that TCP is used over UDP for reliability reasons. The software client has to connect, retrieve and process the obtained data.

Other manufacturers such as ABB use a completely different approach. They communicate through a Software Development Kit (SDK) (*ABB-Developer Center*, n.d.) that should be linked in the source code of the client software. This abstraction layer encloses the commands into SDK methods. The client software is then able to obtain the data through those functions with no extra processing.

Finally, other manufacturers such as Comau, do not have a direct way to connect with the robotic arm. To be able to send or retrieve data, a custom script needs to be written using PDL2 programming language (Vukobratovic, Surdilovic, Ekalo, & Katic, 2009) to create a TCP socket server. This script is then executed in the background when the robotic arm turns on, waiting for incoming connections. From this point onwards, the communication can be treated as in the case exposed with Universal robots.

For these reasons a custom SDK is designed and implemented to unify all of these communication methods into a single one. Fig. 1 shows the internals of the proposed communication interface. This SDK will later be used by the monitoring local servers to retrieve status information of the robotic arm.

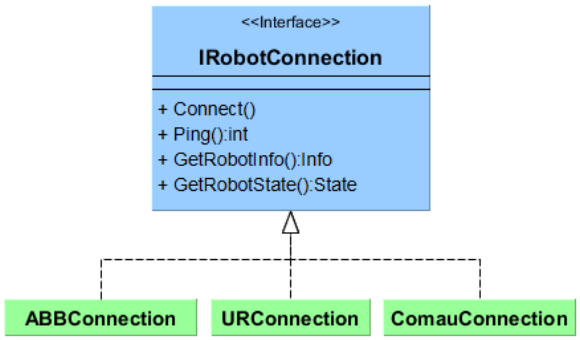


Figure 1. Common interface of the homogeneous connection model.

Each class derived from IRobotConnection is forced to have a minimal set of methods to ensure the homogeneity of the communication. Each subclass manages its own specific way to connect and retrieve data from the desired manufacturer. This is hid-

den from SDK users, as they will use the methods in the IRobotConnection interface for all the different manufacturers and models.

4.2. *Connection protocol*

A custom protocol has been designed to allow data retrieval from the monitoring system and data replay record. As real-time and low bandwidth are both desirable characteristics, the protocol uses raw binary streams to manage data and is message based. The protocol is shown in Fig. 2 to ensure the reproducibility of the proposed method.

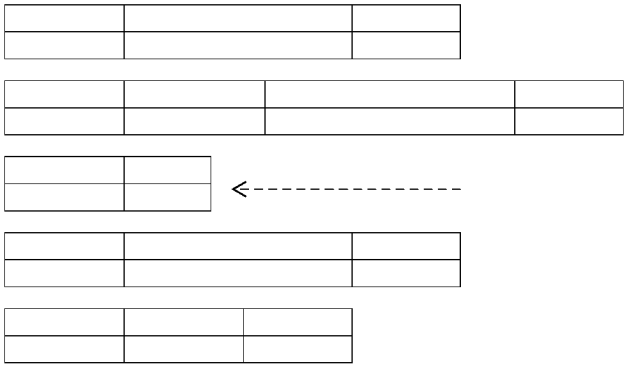


Figure 2. Communication protocol

The first 7 bits of the message allows the system to identify what kind of information is contained inside the packet. It allows up to 128 different types of messages. Even if only a small fraction of these types are used, it leaves a chance to enhance the protocol in the future with more messages.

The remaining bit of the first byte is used as a boolean flag. It can be used to mark a joint data message as error, when the robot with those parameters generates a collision or other problems in the cell. This is useful to mark frames in the replay with error, so it is easier for the monitoring client to jump between error marks, and it is not intended to use as a notification system as there is a specific packet for doing this.

At this point, the packet structure changes depending on the type of message stored. In the context of the experiments described in this paper, five different message types exist.

The first one is the MSG_TIME, containing the time-stamp value of current time inside a 64 bit value. It is sent only once each second to maintain the time line of the monitoring data without too much overhead.

With the second one, MSG_NOTIFY, the system is capable of sending notification messages from the monitoring servers to the clients. It contains a 9 bit value to determine the type of notification it holds. This allows users to adapt the notifications of the monitoring system to their needs, adding up to 512 different types of notifications. It also has an extra 32 bits floating point part to store a value associated with the notification, such as angle, temperature or voltage.

The third one, MSG_DOF, is used to change the number of DoF (Degrees of Freedom) of joint messages containing the robot pose. This way, robots of any number of

DoF can be used inside the monitoring system. This message is only sent at the beginning of the stream to inform the client about the size of MSG_JOINTS messages. If no message of this type is sent, the default value of 6 is used for DoF.

MSG_JOINTS message type is responsible for maintaining the angle values of each joint in the robotic arm chain. Each angle is contained inside a 4 byte single precision floating point number. There is no accumulation of error by using single floating points for joint data, as the system is always asking for the latest joint position of the robotic arm. Also, robot manufacturers state in their manuals that single floating point numbers are used to store joint data, with a range of $\pm 1.18 \times 10^{-38}$ to $\pm 3.4 \times 10^{38}$. After a MSG_TIME message is sent to mark the time-stamp of the stream, several MSG_JOINTS messages are sent at a rate of 30 Hz to describe the actual movement of the robotic arm. This data can be used later to reconstruct the movements performed by a robot inside a cell. The rate can be changed by the user without difficulty, but the difference in bandwidth and storage has to be taken into account.

Finally, the MSG_SELECT message allows a client to ask a server for data of a specific robotic arm when there are many of them available. The 32 bit ID parameter identifies each robot in a unique manner on each server.

The simplicity of the protocol is one of its strong points, as it allows to reduce the required bandwidth and disk space storage to the minimum. It also allows to be easily implemented in other platforms or programming languages.

4.3. *Distributed monitoring*

Fig. 3 shows the proposed distributed model for this monitoring system based on the Industry 4.0 approach. Thanks to the characteristic production distribution proper to 4.0, the ability to monitor multiple shoe factories through internet has been added to the model. The final decision to allow users to use the monitoring system remotely or only locally is from the company. To handle this need, every factory must have a monitoring server. Servers are designed to handle multiple concurrent users. These servers are firewall protected to reduce the security risks of servers exposed to internet.

With the servers designed to be read-only, the risks of a security breach are reduced. The monitoring system only allows requests of information about the state, but in no case it is able to modify the position or state of the robotic arm inside the monitored cells. If this ability to modify the robot state is introduced, then physical damage to the environment or to the workers may occur. (Haddadin, Albu-Schäffer, & Hirzinger, 2011; Bicchi, Peshkin, & Colgate, 2008)

However, the stream of data in the monitoring system should be protected from unwanted attackers. It may be used to spy the current factory production, or any problem in the machinery.

In order to protect the data, all the layers of connection are password protected. In addition, a firewall allowing only the required traffic and avoiding the rest, helps to increase the level of security.

Servers wait for incoming connections in a TCP port. To do so, they run a custom server application coded in C# that handle multiple concurrent users. Webservices or server applications coded in other languages such as Java or Erlang can be used to create the server application as long as they handle the connection protocol discussed in this paper. As the proposed system is using TCP/IP, the described method is compatible with wired, WiFi networks and a mix of them. Servers are also constantly

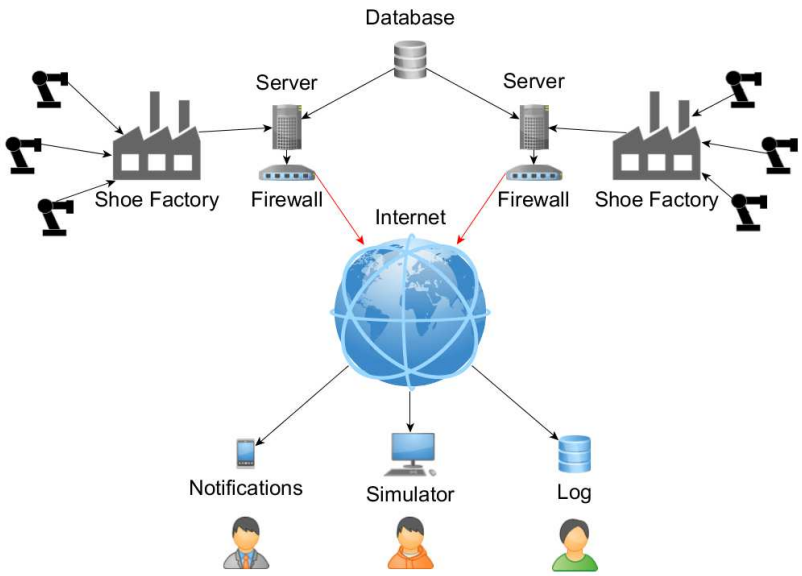


Figure 3. Diagram of the distributed monitoring system's.

polling status data from every monitored robotic cell of its network using the homogeneous cell communication protocol to maintain an updated cache with the status of each robotic arm. Users are able to send query commands to the server, and it will respond by gathering the requested information from the local cache filled with the polled data. All the data passing between users and servers is encrypted with AES-256 (Miller, Vandome, & McBrewster, 2009), to avoid man-in-the-middle attacks. The use of this symmetric encryption algorithm is required due to the nature of real-time queries needed to maintain a smooth movement inside the 3D simulator software. Also, to be able to start querying the server, the user needs to be validated by checking user credentials against a database. Validation is performed within a secure session opened with the asymmetric encryption algorithm RSA (Jonsson & Kaliski, 2003), allowing the secure distribution of the private key used in the symmetric encryption block cipher.

Even though having physical access to all the hardware in the cells is a security risk, with these steps, security is increased, specially when accessing the system from outside the factory where there is no physical access to them. This is not a novelty approach of cybersecurity, but it is adapted to the specific case of the proposed distributed model, to ensure proper security is achieved.

5. 3D simulator as monitoring client

A monitoring system needs one or more clients to allow the user to watch and analyse the status of the monitored factory.

Apart from the notifications and alerts required in monitoring, it is useful to be able to see a visual representation of individual monitored elements in the footwear factory. In case a failure is notified to the user, or simply to check the production or status of a robotic cell, this capability can be used to check the problem and try to

find its root.

One solution may be to put cameras around every element in the factory to be monitored. This option has many drawbacks, as listed below.

- Cost of the cameras, for each cell
- New wired installation required
- Fixed monitoring angle
- High disk space requirement when saving replays
- Large network bandwidth for video feed
- Big amount of recorded data to check in case of error

The method used a custom 3D simulation software (Román-Ibáñez et al., 2018), which was already present in the footwear factory design. The decision for using it was based on the drawbacks listed for the camera-based option (Song, Ding, Kamal, Farrell, & Roy-Chowdhury, 2011).

This custom software is meant to be an all-in one solution for cell design, simulation and monitoring in footwear factories with automated cells. As it is focused on monitoring, one desired feature is to be able to install and run it on different operating systems. Existing software modules were already coded in a specific programming language, and this led to choose the custom software path. However, other simulation software packages such as ROS (Quigley et al., 2009) and their modules can be used to reproduce the proposed system or part of it if it suits better the needs of the user, as long as the structure and the protocol used in the current paper are followed. With this software there is no need to spend more money in extra hardware, because it simulates the cell movements with the status data obtained from the cells. Furthermore, no extra wired installation was required because the existing 4.0 network was used to fetch data directly from the cells.

With cameras monitoring the robotic cell, users receive a video feed of a fixed position and orientation (Yang, Tham, Wu, & Goh, 2009). In contrast, the described method enables users to move freely in 3D space, covering translation, rotation and zooming in any coordinate. This allows 360º inspection of the robotic cell and becomes especially useful in situations like the one shown in Fig. 4. In this case a collision of the robot against the back side of the protection cage can only be seen in Fig. 4.b by looking at a different camera perspective than in Fig. 4.a. These collisions have been detected in offline to depict the need of a monitoring system for automated robotic cells with dynamic input.

Having a software simulator as a client also allows the user to pause, modify speed and position of the replays. This will be useful when analysing them offline.

In cases where the user wants to save the monitoring session for later offline analysis, less space is required to do so. In the camera-based option, users need to save images while the proposed model only stores a few bytes describing robotic arm joint positions at a specific time. For the same reason, network bandwidth requirements are also lower.

With the possibility to monitor several robotic cells simultaneously, the lower bandwidth overhead may help to maintain the system in real-time and with low latency. It is also possible to increase the number of recorded frames per second or the duration of the replay with lesser impact on available resources.

Having stored several days worth of recorded data becomes an issue when the user needs to look for a problem in production. In order to analyse the root of the problem, the user would need to locate and replay a short interval of time around the timestamp of the error. Having a lot of stored data this can take a lot of time. A solution to this problem is given by the simulation software, taking advantage of the bandwidth

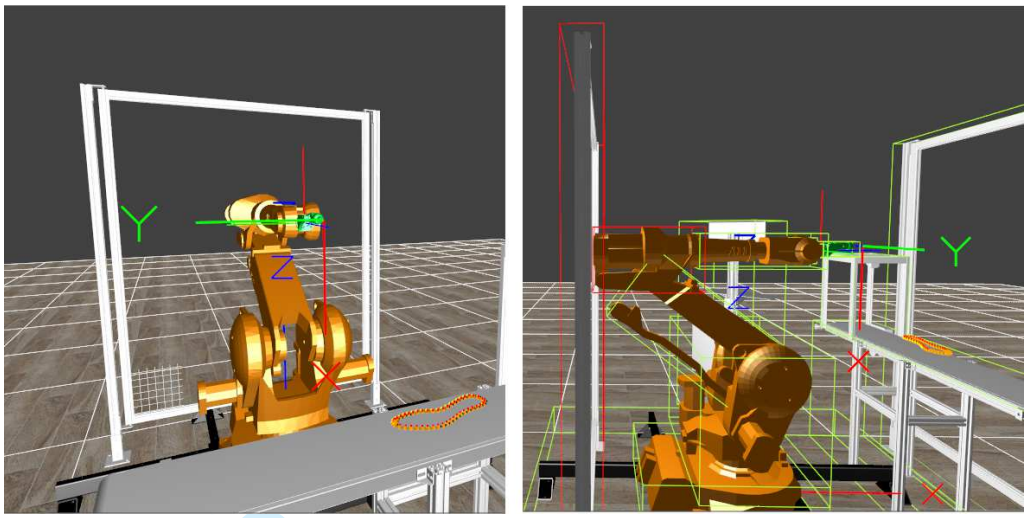


Figure 4. a) Collision hidden by perspective. b) Shown from another camera angle

and space available due to the optimization method described. An error flag is added after each chunk of recorded data with the joint angles of the robot. This error flag is turned on when a problem is detected for that robotic arm position. This way, users may navigate through all recorded errors without having to visually check the whole monitoring session stored.

Another advantage of the simulation software is the ability to use replay analysis to find a solution to a problem or to prevent one. This can be performed by modifying the objects inside the scene while replaying the same movements of the robotic arm inside the cell. This way, one can analyse what would happen to a modification of the physical cell before making any change to the real counterpart for the same movement.

Experiments in this paper will support the claims about feasibility, bandwidth and storage of the proposed system.

The described simulation software is shown in Fig.5.

Each cell needs to be designed in the Graphical User Interface (GUI) of the simulator, with all the required information on the robotic arm used in that cell.

The simulator uses OpenGL (Shreiner, Sellers, Kessenich, Licea-Kane, & Group, 2013) to render the models and the robotic arm of the scene in a three-dimensional environment.

In order to simulate the movement of the robotic arm, forward kinematics were used. They allow the simulator to know the world position and orientation of each joint when rotations are applied to them locally.

The standard Denavit-Hartenberg convention (Radavelli, Simoni, Pieri, & Martins, 2012) has been used to describe the kinematic chain of the robotic arms in the simulator. The parameters of the DH table vary for each model and manufacturer. Each DH matrix ${}^{i-1}T_i$ is capable of converting from coordinate system $i-1$ of a joint into i. Forward kinematics T of the robotic arm are obtained by the successive application of those matrices as shown in Eq. (1), allowing the transformation from an arbitrary joint's coordinate system into another.

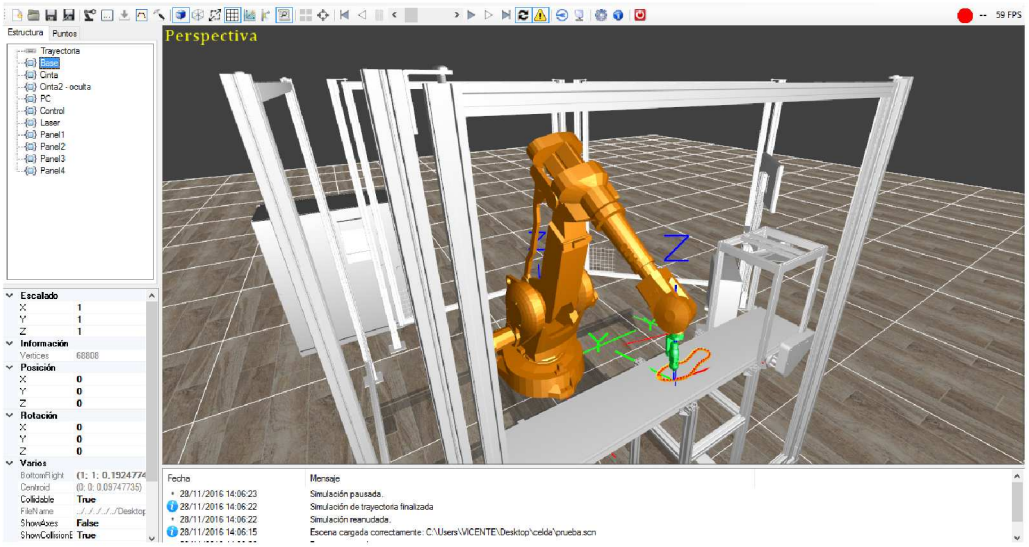


Figure 5. Custom robotic simulator used as monitoring client.

$$T = \prod_{i=1}^{n} {}^{i-1}T_i \tag{1}$$

Using the value of the angles of all the robot joints received thanks to the homogeneous protocol, the software is able to reproduce the actual pose of the real robotic arm. It only needs to apply the forward kinematics of the robotic arm inside the cell to obtain and render the actual pose. There, all transformations allowed by the 3D environment can be applied to the models and the camera to freely inspect the scene from any viewpoint.

The monitoring system maintains a list of each robotic cell of the factory, together with the connection details of each one. This software is constantly polling messages from the cells on the list to retrieve notifications and alerts, as shown in Fig. 6. To obtain more details on the notification received, the user must select the desired cell on the list. The 3D simulator then opens with the loaded associated scene and the robot model in the software starts to copy the same pose as the real connected robot being simulated.

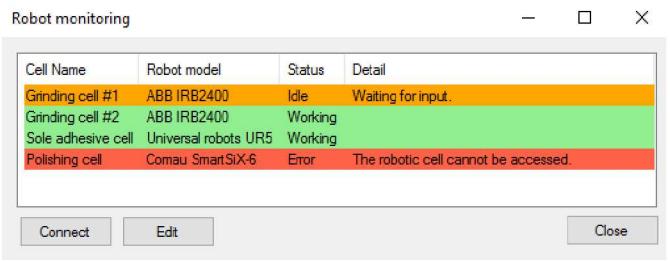


Figure 6. Status of the monitored robotic cells.

However, the proposed model also has some drawbacks, listed below.

- Misalignments between real and simulated world
- Changes in position of physical cell elements.
- Only kinematics are computed, not dynamics.
- The robotic arm is the only monitored motion.

Misalignments may arise due to bad measurements of the real world distances, positions or orientations of each object inside the robotic cell or due to errors while adding the cell inside the software. They will not interfere in the process of the robotic task but may give an imprecise visualisation of the monitored scene.

If elements of the robotic cell environment such as conveyor belt, 3D scanner or the base of robotic arms change their position or orientation after they had been introduced in the software, the monitoring will display an incorrect version of the real robotic cell. This will not happen very often, as once a robotic cell is finished, it will normally not change its design. However, if changes are to be made in the design of the real cell, same changes need to be done in the simulated scene to have both synchronised. This happens because the system gets the values of the robotic arm axes in real-time and use them to calculate the position and orientation of each joint but has no way to fetch or estimate changes in position and orientation for other fixed cell elements.

The proposed model uses only kinematics to simulate the movement of cell elements, focusing on the motion. This does not take into account the interaction of forces between objects or gravity as it is not using dynamics. For the cells encountered in the footwear sector, this has not been a problem and kinematics covered the requirements. This can be a problem in other industries with more complicated interactions of cell elements and may require an upgrade of the proposed model to also use dynamics.

So far, the simulator is only able to get the feed and simulate the movement of robotic arms. The simulation also includes the static 3D models of the rest of cell elements. It has been enough to simulate the robotics cells found in footwear factories. However, the protocol and model can be adjusted to support other hardware components, as long as they provide data feed of the internal status.

Most of the described drawbacks of the proposed system come from the adaptation made to the footwear sector and the possible misalignments between the real and simulated world. If this system is going to be used in other industry sector or application, those drawbacks needs to be taken into account. Future works on the proposed model may reduce the listed drawbacks.

6. Prototype construction and testing

In order to check the viability of the proposed monitoring method, a factory prototype was built. As there are currently no footwear companies sufficiently automated to integrate the proposed system, the experiments carried out in this article have been limited to a laboratory environment.

To obtain a faithful representation of the method in a shoe factory environment, the system was built with several robotic cells. Although in a real scenario cells may perform different tasks, this prototype consisted of cells designed to perform the same task with different approaches. The same experiments can be performed with other kinds of robotic cells. The only reason the selected cells were used in the current study was that they were already available in the laboratory.

Every cell had a different robotic arm in order to check the proposed homogeneous

protocol in the monitoring application across different models and manufacturers. The described prototype is detailed in Fig. 7.

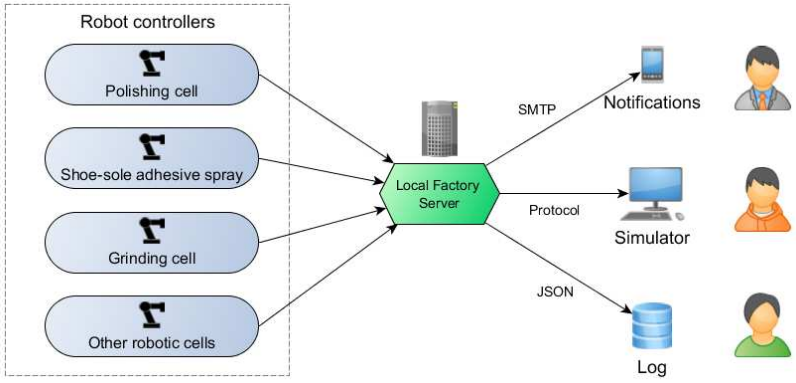


Figure 7. Prototype of the proposed method.

This is a prototype based on a subset sample of the distributed monitoring system already explained in the previous section. It represents a single monitored shoe factory with four robotic cells. In the same network, a local server is constantly fetching data from every robotic cell controller at a fixed interval to maintain a local cache of their statuses. Client can now connect to the local server from internet or from the same internal network with the 3D simulation software and using the custom communication protocol described to view the motion of the robotic arm in real-time or to save the replay of those movements. The server can also save a log of the notifications raised at the controller level in JSON format. It also notifies users using SMTP mail protocol or by using the custom protocol if the simulation software is connected.

6.1. *Prototype cells description*

All the robotic cells were manually recreated inside the simulator, generating a virtual scene for each one. Connection details of each robotic controller were then entered in the monitoring application, including host, username, password and associated scene.

The robotic arm model and manufacturer were included inside the scene, but the connection details were not, because the same scene could be used for another cell with same characteristics.

The first simulated cell, shown in Fig. 8, uses a collaborative Universal Robots UR5 to spray adhesive on top of shoe soles. In this case the paths were precomputed previously offline. Fig. 8 shows both the real cell and the simulated one.

Fig. 9 shows another simulated cell, using an ABB IRB-2400 to apply adhesive to the sole. The path is calculated using a laser scanner on the outer side of the rotating conveyor.

The last cell in Fig. 10, is governed by a Comau SmartSIX robotic arm. It also applies adhesive to the sole, but in this case with hot-melt (Heider, 1999). The cells include a conveyor belt to move the soles from the user to the robotic arm, passing through a laser scanner.

Figure 8. a) Offline adhesive spray of shoe soles. b) Shown in the simulator.

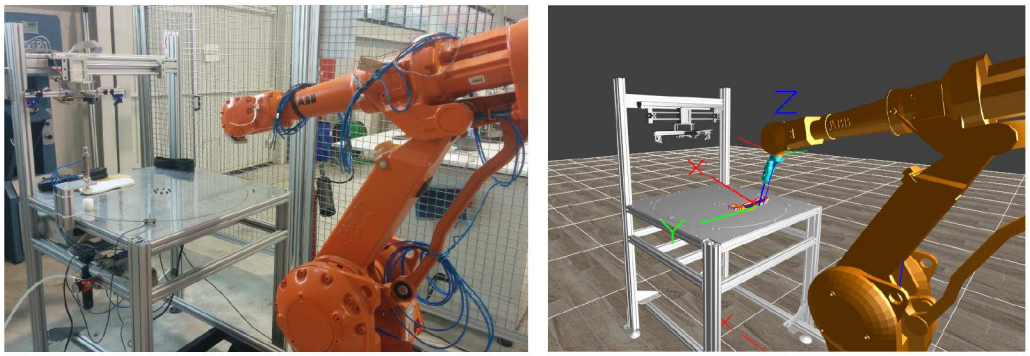


Figure 9. a) Shoe sole application with laser scanner. b) Shown in the simulator.

6.2. *First experiment: Monitoring a collision situation*

Once the prototype was built, an experiment was performed to test the correct behaviour of the proposed monitoring method.

In this case, the robotic cell used to apply adhesive to shoe soles was used as test bench. These cells include a laser scanner module that obtains a 3D representation of the scanned shoe sole. This data was then used to generate a path composed of a set of 3D points with its normal (Chuanyu Wu, Leiying He, Qinchuan Li, & Xudong Hu, 2008). This generation was processed locally inside the cell by software. The robotic arm was then able to apply adhesive using a specific spray tool, following each point and the orientation stated by its normal. The whole process was repeated for each shoe sole introduced inside the cell.

It can happen that an undesired light source such as sunlight, flashes or reflections affect the laser line projected in the conveyor belt. This can lead to undesired 3D representations of the sole, and hence to invalid paths with displaced points and incorrect normals as shown in Fig. 11, where the ray of light is treated as part of the sole. This generates an incorrect trajectory with displaced points. The circled area is zoomed in Fig. 12. This figure shows a detailed view of a part of incorrect normals or point orientations. If a robotic arm follows the twisted orientations, marked as dangerous in the figure, perpendicular to the XY plane and close to the sole, it may collide with the environment.

A safety measure is currently implemented in the robotic cell, avoiding the execution of a path point in case the attached normal surpasses a threshold safety value or when a point is located outside a predefined bounding box. Fig. 13 shows the robotic arm

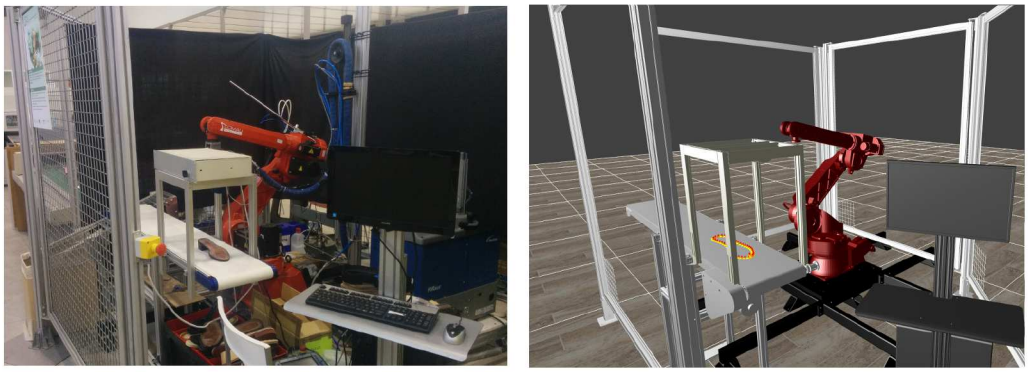


Figure 10. a) Shoe sole application with hot-melt. b) Shown in the simulator.

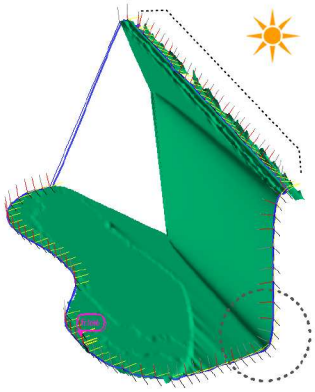


Figure 11. Path generated with incident light during the scanning process.

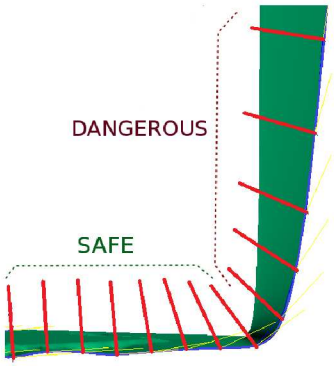


Figure 12. Detailed view of the incorrect normals generated.

stopped at a dangerous normal just when it was about to hit the conveyor belt, after following the path generated with incident light.

Fig. 14 shows the simulator rendering the pose of the monitored scenario by using the data gathered from the physical robotic cell through the homogeneous cell protocol layer. This figure is a static capture of the monitored motion of the robot following the generated trajectory. This motion plus the notification messages received help the user to determine the possible source of the problem. The ability to replay this motion allows the user to review and process it off-line, with useful tools like slow motion, rewind, pause or fast forward.

When this threshold is surpassed, the safety system holds the execution of the path, stopping the robotic arm to avoid damage to the environment or to workers. This halt can withhold the production until a worker verifies that everything is safe and production is resumed with the next sole. The proposed monitoring system can be useful to reduce the time during which production is stopped and therefore increase profit.

In the experiment, the ability of the monitoring system to detect, transmit and store the replay was tested. The sequence of detected communication packets of 34 monitored seconds is illustrated in Fig. 15, showing expected behaviour.

The servers were constantly polling data from the robots to obtain updated data at any time. The user started the login process in the monitor, and it went to the

Figure 13. Robotic arm in a dangerous pose after following an incorrect normal.

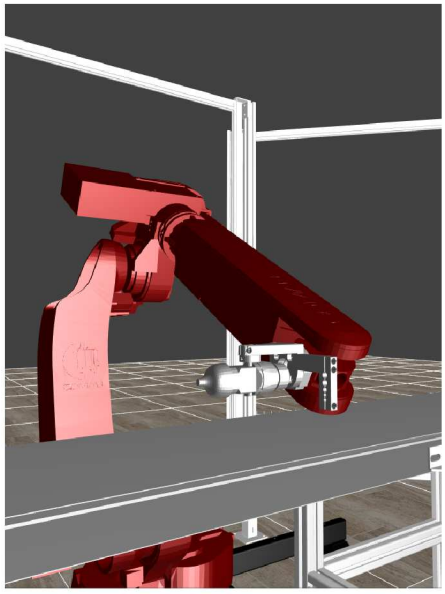


Figure 14. Monitoring the dangerous pose with the simulator in real-time.

server. Once the server responded with a confirmation of the validated login, the user selected the desired robot. As there were not messages to determine the degrees of freedom of the robotic arm, the default of six was used, as stated in the protocol section of this article. The server then responded with one timestamp and a quantity of messages with joint data at a predefined rate. This allowed the user to obtain a representation of the pose of the robotic arm. Once the threshold was surpassed, the server sent a notification message to the monitor to alert the user of a malfunction in the production. Latency was less than 5 ms on each measure, as the server works in the local network where the robotic arms reside.

6.3. *Second experiment: Bandwidth comparison*

The next test compared the required bandwidth of a camera-based monitoring system with the proposed method.

To calculate the bandwidth required by the proposed method several aspects had to be taken into account. The most data intensive part of the protocol was located at the point of retrieval of the angles of all joints in the robotic arm. As specified in the protocol section of this document, each angle was sent over the wire as a single precision floating point number with a size of 4 bytes each in a MSG_JOINTS message. For a typical industrial robotic arm with 6 DoF, this will result in 24 bytes. In addition, with the header and the error flag it will sum 25 bytes for a single frame or 6,000 bps at 30 FPS. Also, every second, a time-stamp packet is sent to synchronize the stream, with a static size of 9 bytes. With a 30 Hz rate, the system needs a total bandwidth of 6,072 bps or 759 Bytes for every second of storage.

Table 1 shows the comparison between different scenarios of camera feed with the proposed method. The reddish lines correspond to various combinations of resolution and compression codec for camera feed. The greenish ones contain the data of the method both in raw and compressed modes. In this case, resolution was not specified

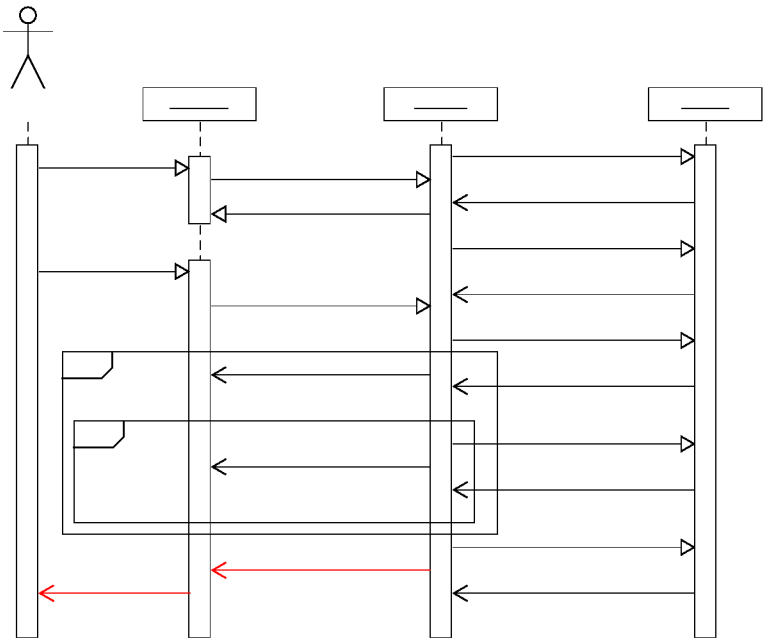


Figure 15. Sequence diagram of the detected messages during the test.

as replays were based solely on movement data. It could be set to any resolution supported by the computer running the simulator, without any impact on bandwidth or disk space.

Table 1. Bandwidth and storage space comparison between proposed method and camera feed.

Resolution	Codec	FPS	Bandwidth	Storage@day
480p	MJPEG	30	7.2 Mbps	544.3 GB
480p	H.264	30	1.4 Mbps	108.9 GB
720p	MJPEG	30	19.2 Mbps	1,500 GB
720p	H.264	30	3.8 Mbps	290.3 GB
–	RAW	30	5.625 Kbps	65.6 MB
–	ZIP	30	3.626 Kbps	42.3 MB

At the end of a full day, only 65.5776 MB of disk space to save all the movements for future replays of a single robotic arm were required. This requirement was 1,699 times lower than the best codec tested with camera feed. These results were for raw data, but the resulting data file could be compressed to reduce the disk space it needed even more. A fast test with Zip on hour long recorded files gave an average compression ratio of 1.55:1, and a resulting average file of 1.76 MB. This time, the gain with respect to the camera feed rose to 2,635 times. Another compression test performed with a replay of a robotic cell performing the same trajectories during the same period gave a final compressed file of only 68 KB. The ratio in this case was as high as 39.11:1. A final test with full random trajectory points gave a resulting file of 2.51 MB with a

ratio of 1.1:1.

This experiment measured the reduction in storage space and bandwidth both in raw and compressed configurations of the proposed method compared to camera feed of different resolutions and compression codecs. It also showed that compression ratios can vary depending on data repeatability inside the recorded stream.

6.4. *Third experiment: Network Stress Test*

In this experiment, the limits of the networking capabilities of the proposed model has been tested. The proposed model needs to support a low number of concurrent users, but each querying a high number of robotic arms.

To perform this experiment, several virtual clients were spawned to simulate a high quantity of robotic arms without the need to have them physically. Erlang has been used to manage such concurrency levels (Armstrong, 2007). Each node will answer random pose data to the local server that queries them. The local server will be constantly polling uncompressed data from each node as if they were real robots. This is intended to stress the local network of a factory.

Fig. 16 shows the resulting graph after stressing the local network with an increasing number of robotic virtual nodes and queries from one local server simulating a single factory. It shows that the proposed model uses only a small fraction of the local network bandwidth and can afford the monitoring of a high quantity of robotic arms, more than enough to cover any factory. This have a minimal impact on any modern network capable of speeds from 100 Mbps onwards. It can even handle up to one thousand robots with an old 10 Mbps network. This leaves room in the local network to other communication packages.

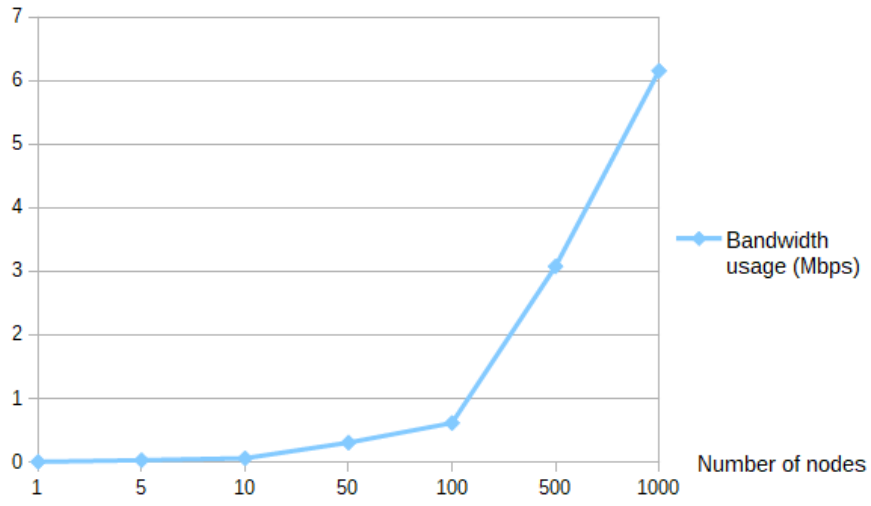


Figure 16. Bandwidth usage of the local network stress test.

The proposed method also has the ability to monitor the robotic cells remotely through internet. This requires the local server located at each factory to support concurrent users. The system is designed so the bandwidth usage of each local server will not raise on each new concurrent user connection of the local server. This was achieved by maintaining a local cache of status data from each robot inside each local

server. This way each server maintains the data of the robots of his factory. With this approach, the server is constantly polling status data from its robots. The experiment shown that this will not be a problem. When a user queries the server, it responds with the latest copy of status data it has stored, without requiring extra queries to the robotic arm. There will not be a high number of concurrent users querying the servers, so concurrency is not a problem as long as it is present. If needed, redundant local servers can be added to handle the extra concurrency needs. Also, if a remote client has connection problems due latency or interruptions in the service, data is still stored in the server with timestamps to ensure the correct order, and a reconnection of the remote client to the server when networking problems disappear will give the correct data.

The results of this experiment prove that the network capabilities of the proposed model are higher than the required in real factory scenarios.

7. Conclusion

A new industrial revolution called Industry 4.0 is taking place before our eyes. This revolution is changing how manufacturing is implemented in the industry, enhancing and optimizing the production of goods. It represents the next step in industries such as footwear, where shoe production is highly distributed across several factories, each dedicated to making a different part of the final product. Those who do not adjust to this revolution may become outdated and one step behind the rest.

This dispersed distribution of shoe making factories explains the need for monitoring. Obtaining real-time information from each shoe factory involved in the production process is essential to get a snapshot of the current status of the whole production process. It may also facilitate the decision making process and notify the workers when an error occurs, increasing the uptime of the production line.

In shoe factories, automation is performed using cells, with robotic arms and specific tools attached to them. A homogeneous connection module was designed and implemented to allow the monitoring system to fetch data from different robotic arm models and manufacturers.

Security was another factor taken into account to reduce the risk of information leaks. These can be dangerous as third parties can steal data and gain an edge over the spied competitor. Existing encryption algorithms were applied to data streams. Additionally, a requirement to log into the system to gather information was added. Both measures reduced the exposure to unwanted intrusions. Servers were designed to be read-only, to prevent unwanted modifications of robotic arms joints that could lead to physical damage to the factory environment and to their workers.

Having a 3D simulator as a client allows 360º inspection of the monitored robotic cell. The proposed model works in real-time, but it allows replay storage for later inspection. Experiments performed confirmed that lower bandwidth was required for data transmission as well as less disk space for replay storage, compared to other monitoring solutions such as setups based on standard cameras.

Since the footwear sector is largely made up of SMEs, keeping down costs was considered an important factor when implementing enhancements in production. Using the current 4.0 network and software simulator reduced the cost of implementation of the monitoring system, as well as reusing the software for cell design, simulation and monitoring.

Experiments confirmed the viability of the proposed method. In addition, they

showed that the method offered the desired scalability and was also compatible with other safety systems in the factory.

For these reasons, a software simulator was selected as a monitoring solution for 4.0 shoe factories. It can be upgraded or adapted to other industries with similar characteristics and dynamic automated processes such as food, furniture, toy and aerospace.

Future work derived from this research will aim to enhancing the proposed monitoring solution by minimising the impact of the listed drawbacks described in this paper. Another future improvement is to enhance the monitoring client with the inclusion of immersive virtual reality, allowing a more natural and realistic way to inspect the monitored robotic cells. It is expected that in future shoe factories increase the automation of their processes to a point where the proposed method can be applied and tested outside the test-bench.

References

ABB-Developer Center. (n.d.). Retrieved from `http://developercenter.robotstudio.com/pcsdk/api{\_}reference`

Armstrong, J. (2007). *Programming Erlang: Software for a Concurrent World*. Pragmatic Bookshelf.

Bicchi, A., Peshkin, M. A., & Colgate, J. E. (2008). Safety for Physical Human-Robot Interaction. In *Springer handbook of robotics* (pp. 1335–1348). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from `http://link.springer.com/10.1007/978-3-540-30301-5{\_}58`

Bogue, R. (2009). The role of robots in the food industry: A review. *Industrial Robot*.

Brusaferri, A., Ballarino, A., & Carpanzano, E. (2011). Reconfigurable knowledge-based control solutions for responsive manufacturing systems. *Studies in Informatics and Control*, *20*(1), 31–42.

Carpanzano, E., & Jovane, F. (2007). Advanced Automation Solutions for Future Adaptive Factories. *CIRP Annals - Manufacturing Technology*.

Chuanyu Wu, Leiying He, Qinchuan Li, & Xudong Hu. (2008). Research on the generation of trajectory for shoe upper spraying based on structured light. In *2008 ieee international conference on industrial technology* (pp. 1–5). IEEE. Retrieved from `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4608571`

Conte, G., Facchino, I., Marziali, M., & Mazzocchetti, L. (2013, nov). An innovation process in the marking of lasts for the footwear industry. *The International Journal of Advanced Manufacturing Technology*, *69*(5-8), 1605–1617. Retrieved from `http://link.springer.com/10.1007/s00170-013-5058-y`

Davia, M., Jimeno-Morenilla, A., & Salas, F. (2013). Footwear bio-modelling: An industrial approach. *CAD Computer Aided Design*, *45*(12), 1575–1590.

Fawaz, K., Merzouki, R., & Ould-Bouamama, B. (2009). Model based real time monitoring for collision detection of an industrial robot. *Mechatronics*, *19*(5), 695–704. Retrieved from `http://linkinghub.elsevier.com/retrieve/pii/S0957415809000348`

Gilchrist, A. (2016). Introducing Industry 4.0. In *Industry 4.0* (pp. 195–215). Berkeley, CA: Apress. Retrieved from `https://link-springer-com.proxy.lib.chalmers.se/content/pdf/10.1007{\%}2F978-1-4842-2047-4{\_}13`

.pdf{\%}0Ahttp://link.springer.com/10.1007/978-1-4842-2047-4{_}13http://link.springer.com/10.1007/978-1-4842-2047-4{_}13

Gorecky, D., Schmitt, M., Loskyll, M., & Zuhlke, D. (2014, jul). Human-machine-interaction in the industry 4.0 era. In *2014 12th ieee international conference on industrial informatics (indin)* (pp. 289–294). IEEE. Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6945523

Gracia, L., Perez-Vidal, C., Mronga, D., de Paco, J.-M., Azorin, J.-M., & de Gea, J. (2017). Robotic manipulation for the shoe-packaging process. *The International Journal of Advanced Manufacturing Technology*, 1–15. Retrieved from http://dx.doi.org/10.1007/s00170-017-0212-6

Gungor, V., & Hancke, G. (2009). Industrial Wireless Sensor Networks: Challenges, Design Principles, and Technical Approaches. *IEEE Trans. Industrial Electronics*.

Haddadin, S., Albu-Schäffer, A., & Hirzinger, G. (2011). Safe Physical Human-Robot Interaction: Measurements, Analysis & New Insights. *Robotics Research*, *66*, 395–407. Retrieved from http://www.springerlink.com/content/e5247n1245331486/{\$}{\%}5C{\$}nhttp://www.springerlink.com/index/10.1007/978-3-642-14743-2

Heider, R. (1999). *Moisture-curing polyurethane hot-melt adhesive.* Google Patents. Retrieved from https://www.google.com/patents/US5932680

Hsing, Y. T. (1999). Trading companies in Taiwan's fashion shoe networks. *Journal of International Economics*.

Jatta, F., Zanoni, L., Fassi, I., & Negri, S. (2004, oct). A roughing/cementing robotic cell for custom made shoe manufacture. *International Journal of Computer Integrated Manufacturing*, *17*(7), 645–652. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/095112042000273212

Jazdi, N. (2014, may). Cyber physical systems in the context of Industry 4.0. In *2014 ieee international conference on automation, quality and testing, robotics* (pp. 1–4). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6857843http://ieeexplore.ieee.org/document/6857843/

Jonsson, J., & Kaliski, B. (2003). *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1.* United States: RFC Editor.

Kargermann, B. M., & Achatz, R. (2010). Agenda Cyber Physical Systems: Outlines of a new Research Domain. *acatech, Berlin, Germany*.

Larsson, S., & Kjellander, J. A. P. (2006). Motion control and data capturing for laser scanning with an industrial robot. *Robotics and Autonomous Systems*.

Leo H. Chiang, MS -Richard D. Braatz, P. (2005). *Fault detection and diagnosis in industrial systems*.

Makris, S., Michalos, G., & Chryssolouris, G. (2012). Virtual commissioning of an assembly cell with cooperating robots. *Advances in Decision Sciences*, *2012*.

Mantha, B. R., Menassa, C. C., & Kamat, V. R. (2018). Robotic data collection and simulation for evaluation of building retrofit performance. *Automation in Construction*, *92*, 88–102.

Maurtua, I., Goenaga, I., & Tellaeche, A. (2012). Robotic Solutions for Footwear Industry. *IEEE*, 2–5.

Michalos, G., Kaltsoukalas, K., Aivaliotis, P., Sipsas, P., Sardelis, A., & Chryssolouris, G. (2014). Design and simulation of assembly systems with mobile robots. *CIRP Annals*, *63*(1), 181–184. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S000785061400105X

Michalos, G., Makris, S., Aivaliotis, P., Matthaiakis, S., Sardelis, A., & Chryssolouris, G. (2015). Autonomous production systems using open architectures and mobile robotic structures. In *Procedia cirp* (Vol. 28, pp. 119–124).

Miller, F. P., Vandome, A. F., & McBrewster, J. (2009). *Advanced Encryption Standard*. Alpha Press.

Mourtzis, D., Papakostas, N., Mavrikios, D., Makris, S., & Alexopoulos, K. (2015). The role of simulation in digital manufacturing: Applications and outlook. In *International journal of computer integrated manufacturing* (Vol. 28, pp. 3–24).

Nemec, B., Lenart, B., & Zlajpah, L. (2003). Automation of lasting operation in shoe production industry. In *Ieee international conference on industrial technology, 2003* (pp. 462–465). IEEE. Retrieved from `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1290363http://ieeexplore.ieee.org/document/1290363/`

Pedrocchi, N., Villagrossi, E., Cenati, C., & Tosatti, L. M. (2015). Design of fuzzy logic controller of industrial robot for roughing the uppers of fashion shoes. *International Journal of Advanced Manufacturing Technology*, *77*(5-8), 939–953.

Perez, F., Irisarri, E., Orive, D., Marcos, M., & Estevez, E. (2015, sep). A CPPS Architecture approach for Industry 4.0. In *2015 ieee 20th conference on emerging technologies & factory automation (etfa)* (Vol. 2015-Octob, pp. 1–4). IEEE. Retrieved from `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7301606`

Perez-Vidal, C., Gracia, L., de Paco, J. M., Wirkus, M., Azorin, J. M., & de Gea, J. (2018, feb). Automation of product packaging for industrial applications. *International Journal of Computer Integrated Manufacturing*, *31*(2), 129–137. Retrieved from `https://www.tandfonline.com/doi/full/10.1080/0951192X.2017.1369165`

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., ... Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In *Icra workshop on open source software.*

Radavelli, L., Simoni, R., Pieri, E. D., & Martins, D. (2012). A Comparative Study of the Kinematics of Robots Manipulators by Denavit-Hartenberg and Dual Quaternion. *Mecánica Computacional, Multi-Body ...*, *XXXI*, 13–16. Retrieved from `http://scholar.google.com/scholar?hl=en{\&}btnG=Search{\&}q=intitle:A+COMPARATIVE+STUDY+OF+THE+KINEMATICS+OF+ROBOTS+MANIPULATORS+BY+DENAVIT-HARTENBERG+AND+DUAL+quaternion{\#}2`

Román-Ibáñez, V., Pujol-López, F. A., Mora-Mora, H., Pertegal-Felices, M. L., & Jimeno-Morenilla, A. (2018). A low-cost immersive virtual reality system for teaching robotic manipulators programming. *Sustainability (Switzerland)*.

Schlechtendahl, J., Keinert, M., Kretschmer, F., Lechler, A., & Verl, A. (2015, feb). Making existing production systems Industry 4.0-ready. *Production Engineering*, *9*(1), 143–148. Retrieved from `http://link.springer.com/10.1007/s11740-014-0586-3`

Segal, H. P., & Thomson, R. (1990). *The Path to Mechanized Shoe Production in the United States.* (Vol. 18; UNC Press Books, Ed.) (No. 3). UNC Press Books. Retrieved from `http://www.jstor.org/stable/2702669?origin=crossref{\%}5Cnpapers2://publication/doi/10.2307/2702669`

Shreiner, D., Sellers, G., Kessenich, J. M., Licea-Kane, B., & Group, K. O. A. R. B. W. (2013). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.3*. Addison-Wesley. Retrieved from `https://books.google.es/`

books?id=gnxUewAACAAJ

Song, B., Ding, C., Kamal, A. T., Farrell, J. A., & Roy-Chowdhury, A. K. (2011). Distributed camera networks. *IEEE Signal Processing Magazine*.

Summers, M. (2005). Robot capability test and development of industrial robot positioning system for the aerospace industry. *SAE transactions*.

Uhlemann, T. H., Lehmann, C., & Steinhilper, R. (2017). The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0. In *Procedia cirp* (Vol. 61, pp. 335–340).

Uhlemann, T. H., Schock, C., Lehmann, C., Freiberger, S., & Steinhilper, R. (2017). The Digital Twin: Demonstrating the Potential of Real Time Data Acquisition in Production Systems. *Procedia Manufacturing*, *9*, 113–120. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S2351978917301610

Universal Robots RTDE. (n.d.). Retrieved from https://www.universal-robots.com/how-tos-and-faqs/how-to/ur-how-tos/real-time-data-exchange-rtde-guide-22229/

Vincze, M., & Pichler, A. (2002). Automatic robotic spray painting of low volume high variant parts. *... on Robotics)*.

Vukobratovic, M., Surdilovic, D., Ekalo, Y., & Katic, D. (2009). Control of Robots in Contact Tasks: A Survey. In *Dynamics and robust control of robot-environment interaction* (pp. 1–67). World Scientific.

Wahlster, W. (2012). *From Industry 1.0 to Industry 4.0: Towards the 4th Industrial Revolution.* Forum Business meets Research.

Wollschlaeger, M., Sauter, T., & Jasperneite, J. (2017). The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine*, *11*(1), 17–27.

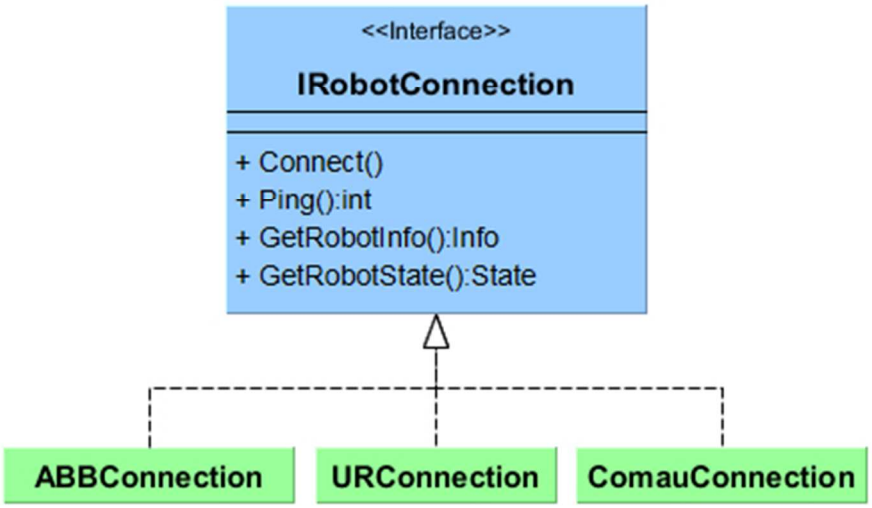
Wu, C. (2008). Research on the generation of trajectory for shoe upper spraying based on structured light. In *2008 ieee international conference on industrial technology* (pp. 1–5). Retrieved from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4608571

Xiong, S., Zhao, J., Jiang, Z., & Dong, M. (2010). A computer-aided design system for foot-feature-based shoe last customization. *The International Journal of Advanced Manufacturing Technology*, *46*(1-4), 11–19. Retrieved from http://link.springer.com/10.1007/s00170-009-2087-7

Yang, M. J., Tham, J. Y., Wu, D., & Goh, K. H. (2009). Cost effective IP camera for video surveillance. In *2009 4th ieee conference on industrial electronics and applications, iciea 2009.*

Zlajpah, L. (2008). Simulation in robotics. *Mathematics and Computers in Simulation*, *79*(4), 879–897. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0378475408001183

Zlajpah, L., & Nemec, B. (2008). Robotic cell for custom finishing operations. *International Journal of Computer Integrated Manufacturing*, *21*(1), 33–42. Retrieved from http://www-hcr.ijs.si/resources/papers/ijcim08.pdf



Common interface of the homogeneous connection model.

162x98mm (72 x 72 DPI)

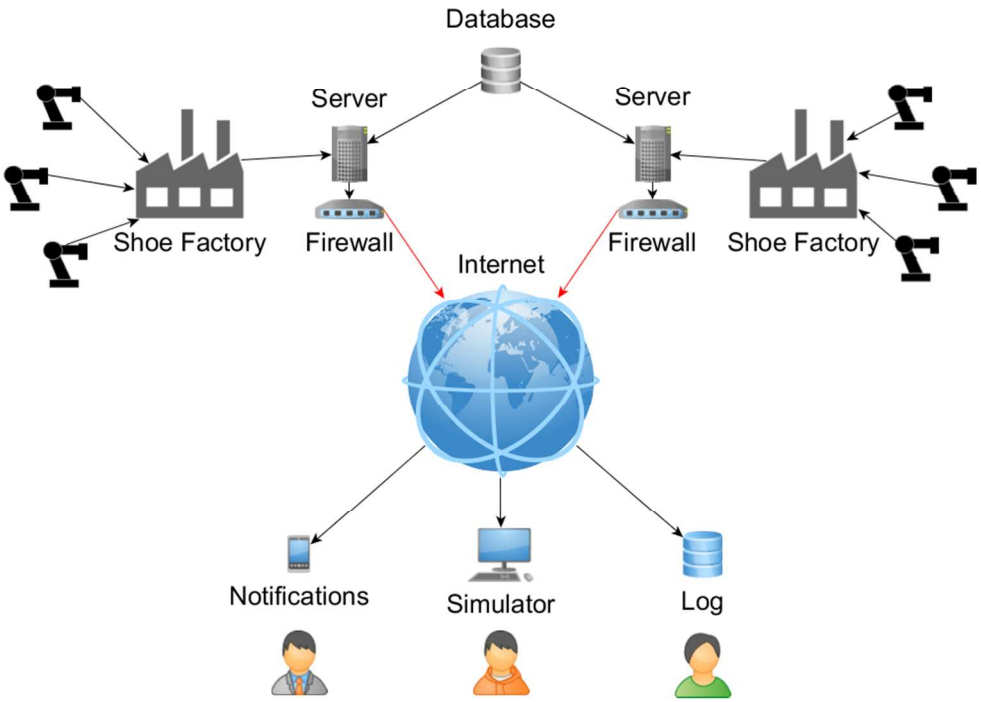


Diagram of the distributed monitoring system's.

357x263mm (72 x 72 DPI)

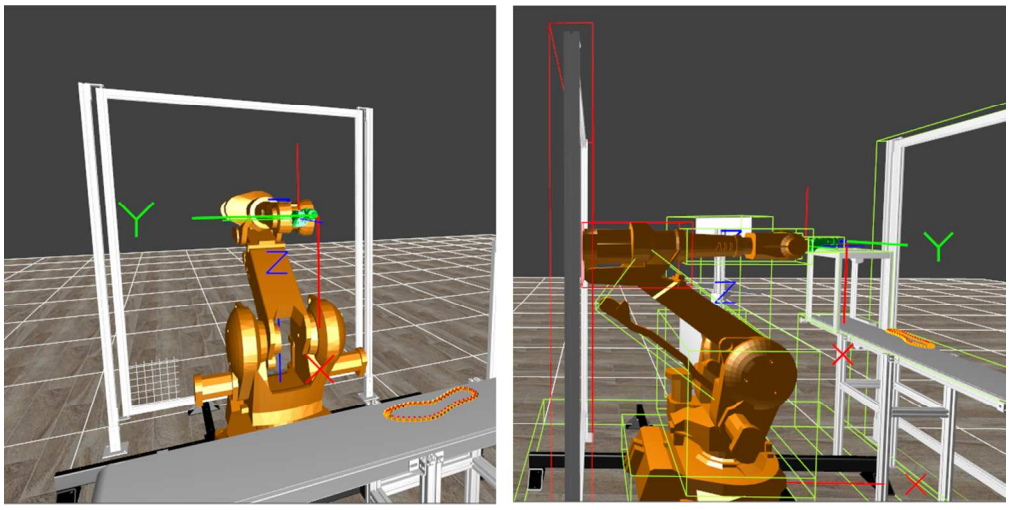
7 bit	64 bit	1 bit
MSG_TIME	Timestamp	Error flag

7 bit	9 bit	64 bit	32 bit
MSG_NOTIFY	MSG_TYPE	Timestamp	Value

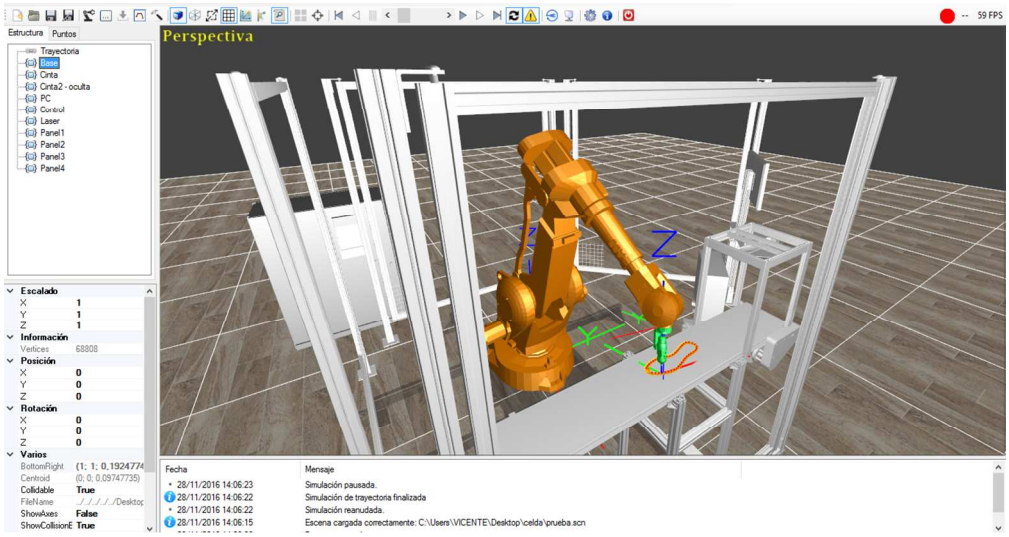
7 bit	9 bit
MSG_DOF	DoF

7 bit	192 bit (6 x 32 bit)	1 bit
MSG_JOINTS	6 x Joint Data	Error flag

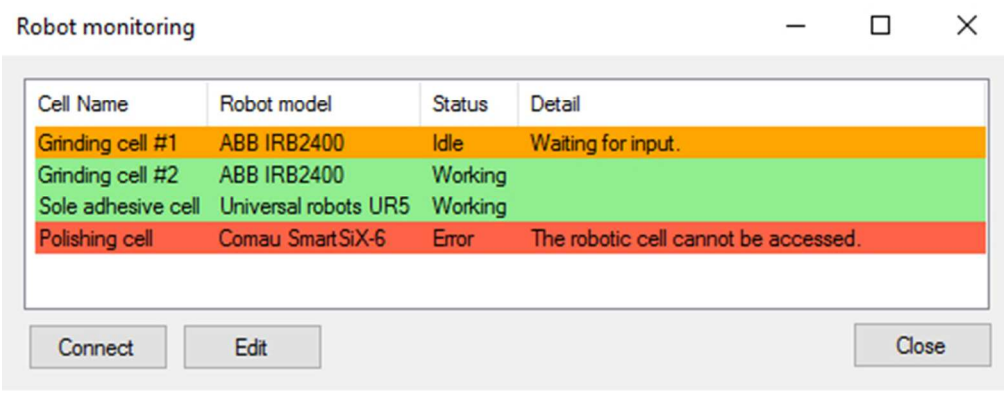
7 bit	32 bit	1 bit
MSG_SELECT	Robot ID	Unused



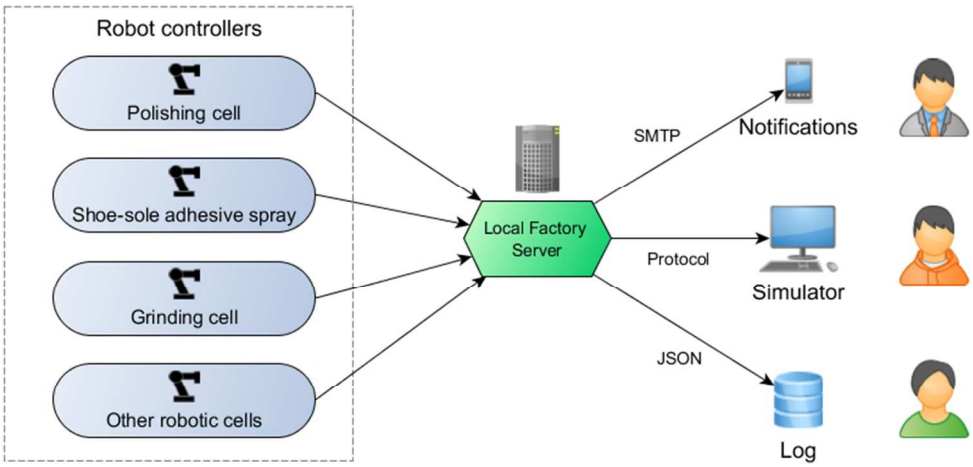
a) Collision hidden by perspective. b) Shown from another camera angle



Custom robotic simulator used as monitoring client.



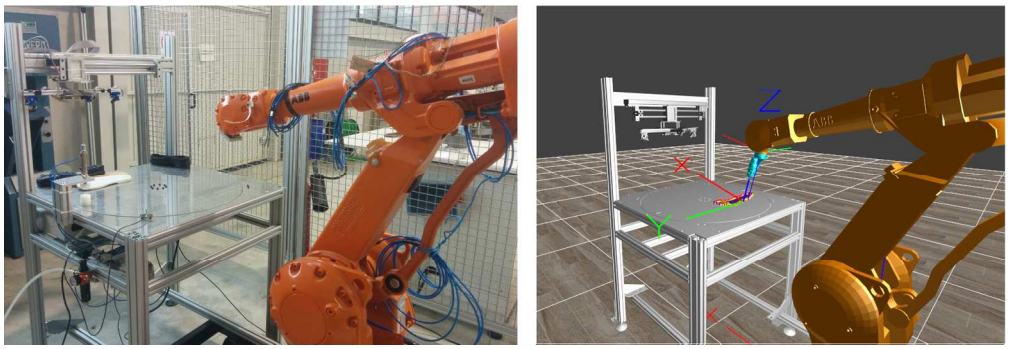
Status of the monitored robotic cells.



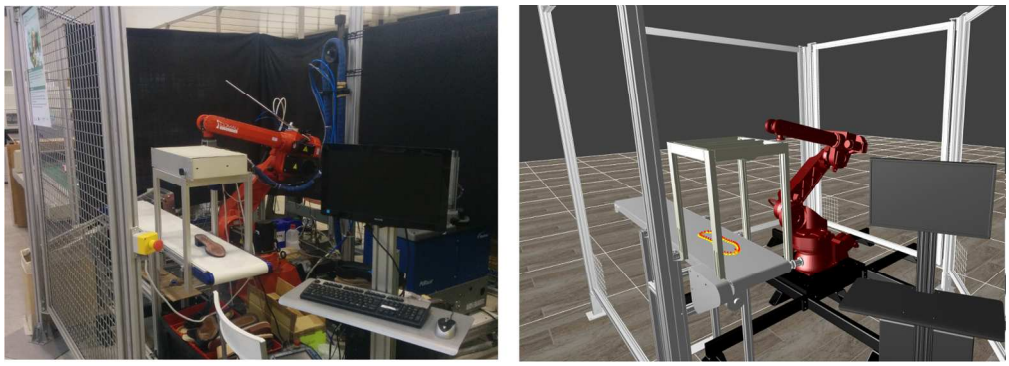
Prototype of the proposed method.

289x143mm (72 x 72 DPI)

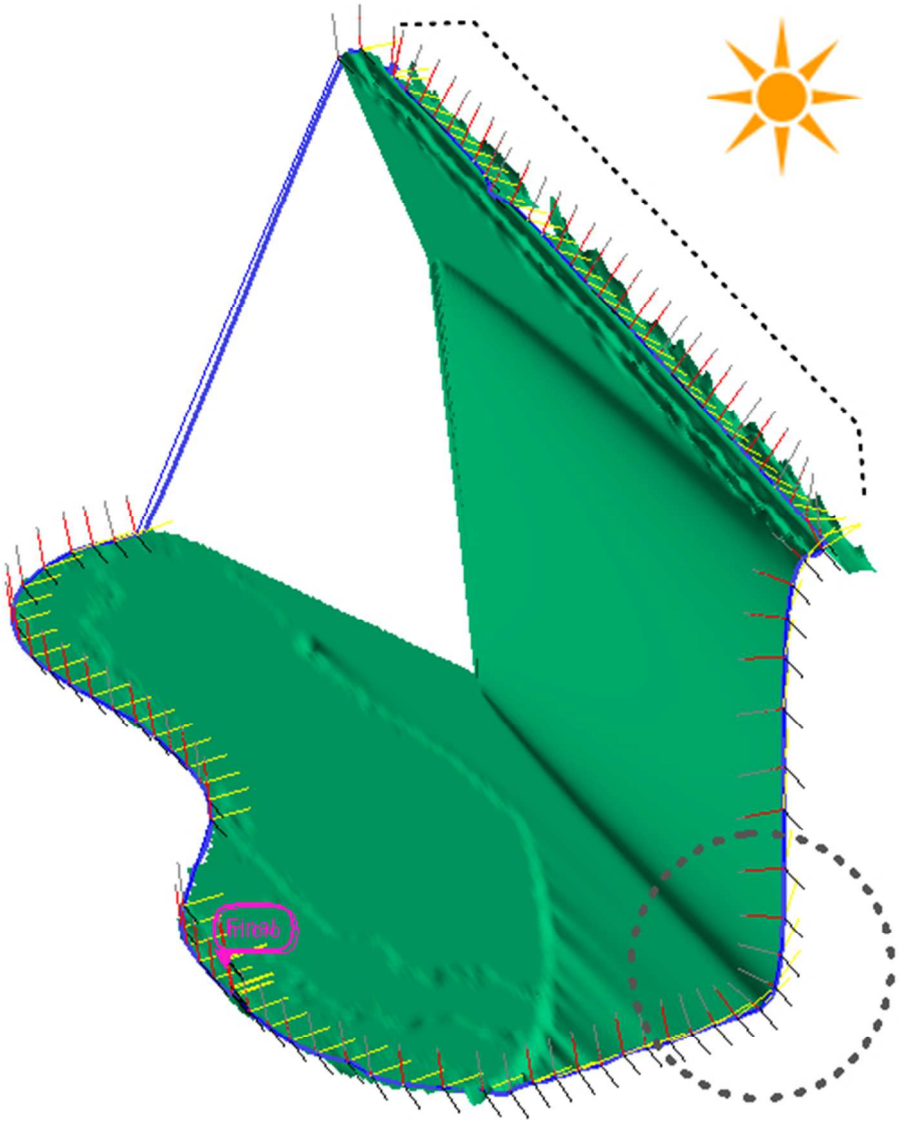
a) Offline adhesive spray of shoe soles. b) Shown in the simulator.



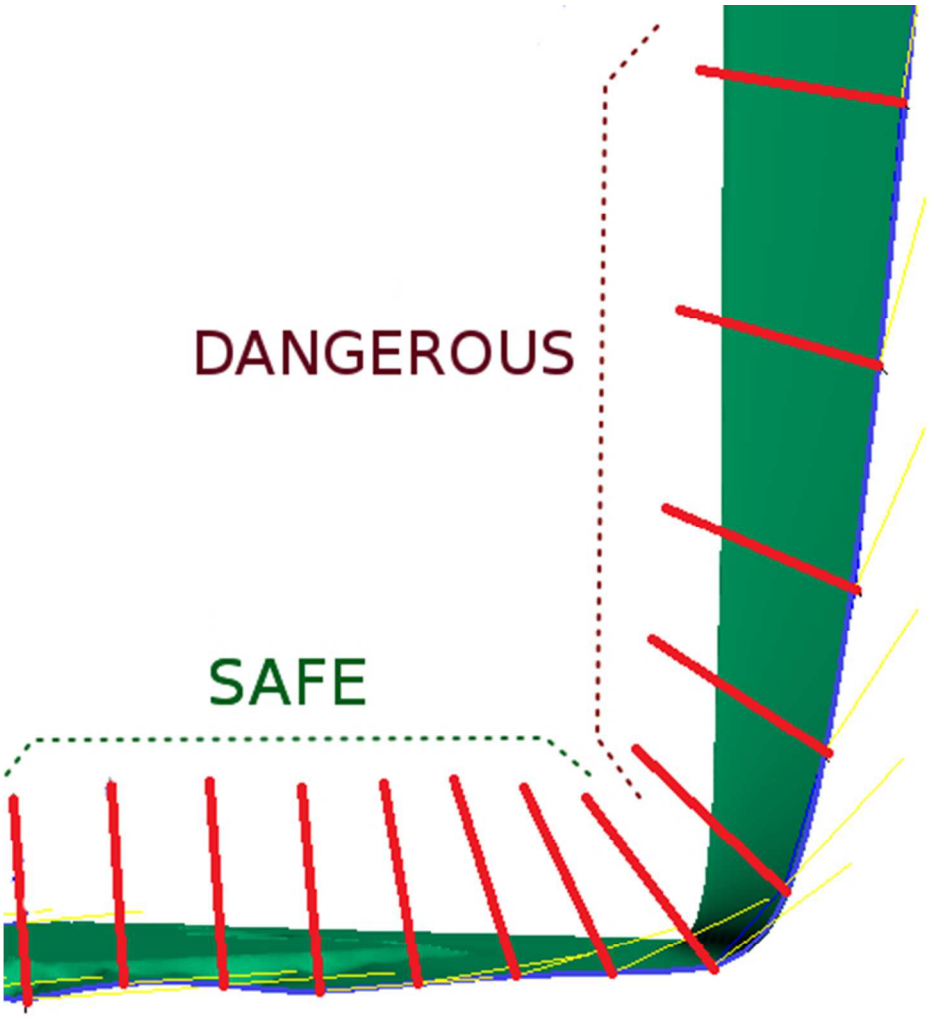
a) Shoe sole application with laser scanner. b) Shown in the simulator.



a) Shoe sole application with hot-melt. b) Shown in the simulator.

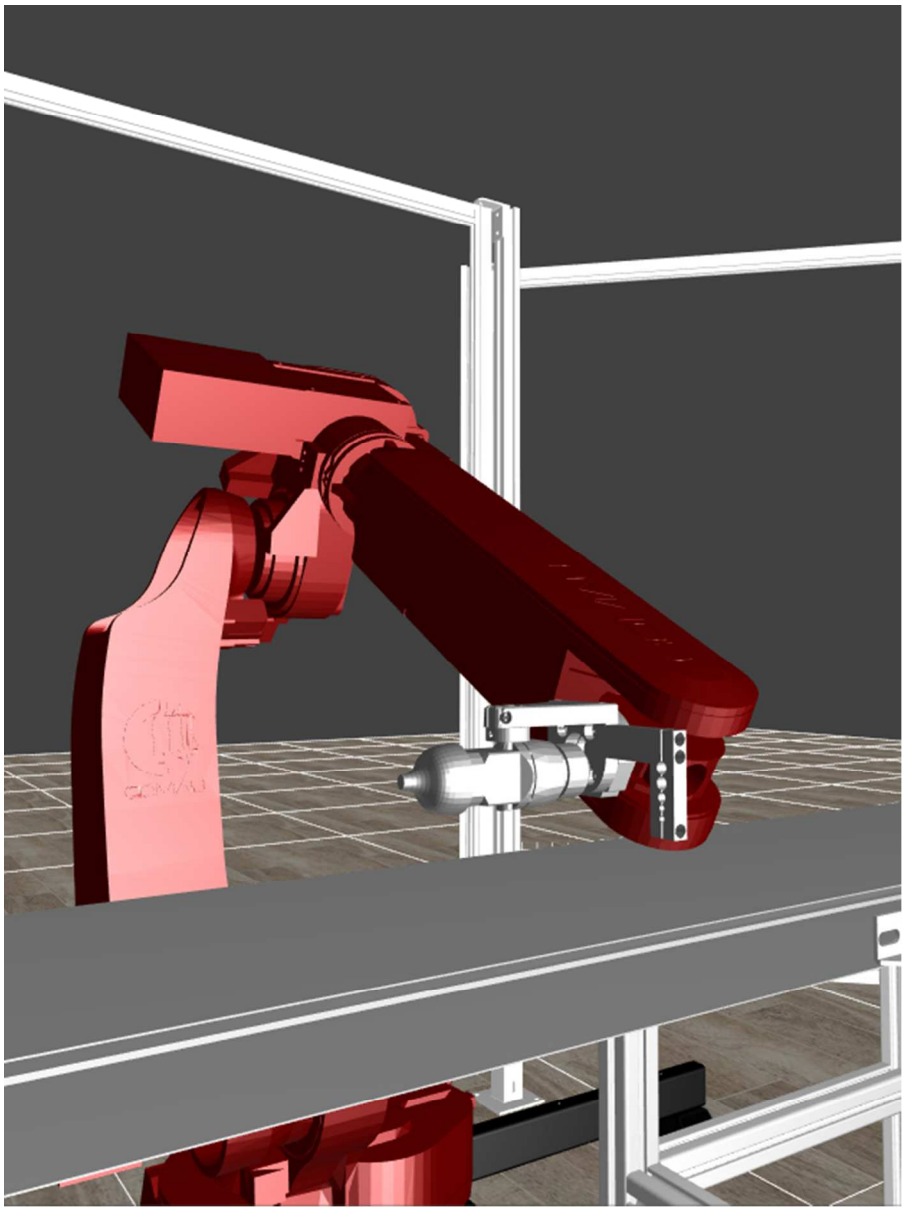


Path generated with incident light during the scanning process.

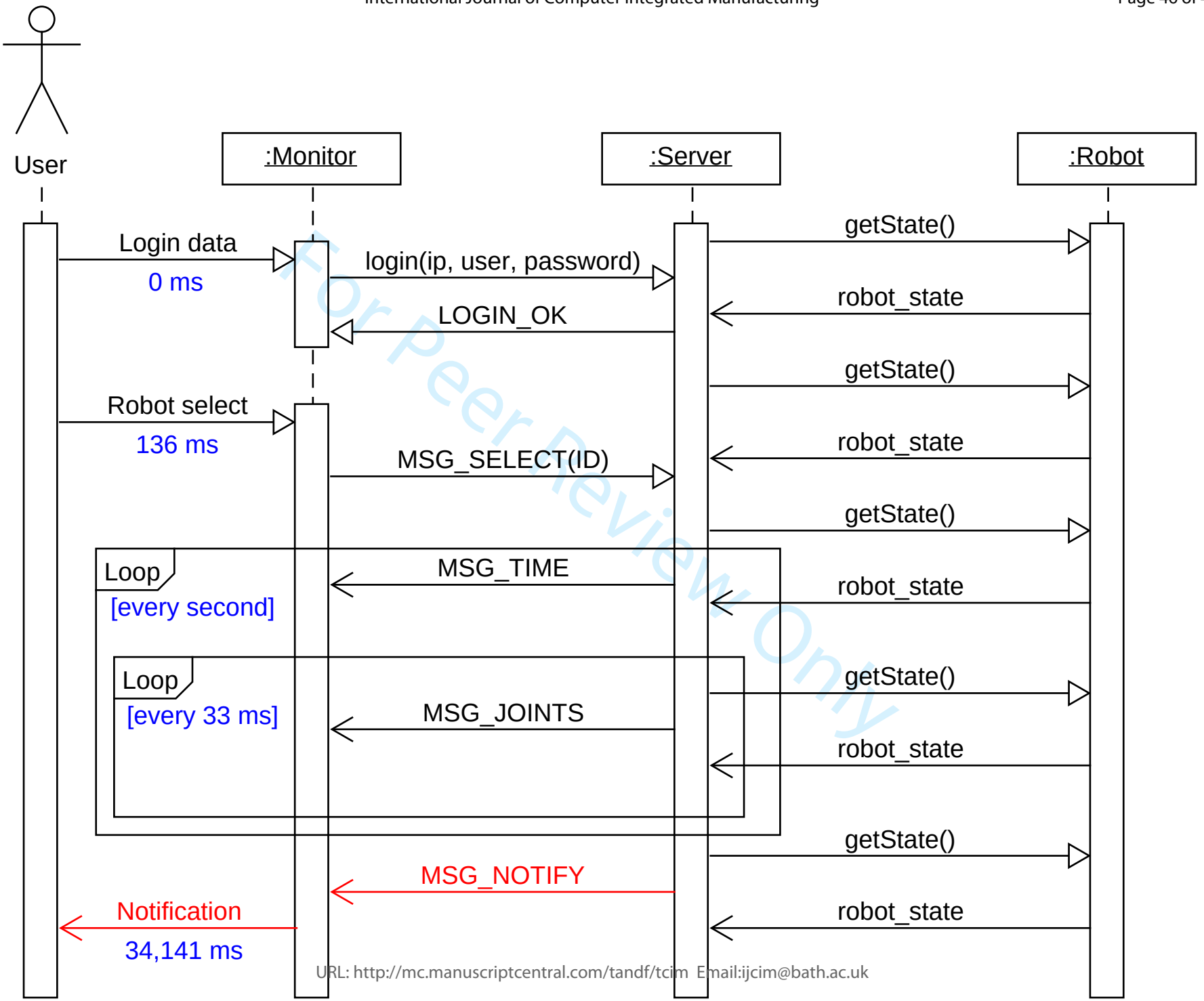


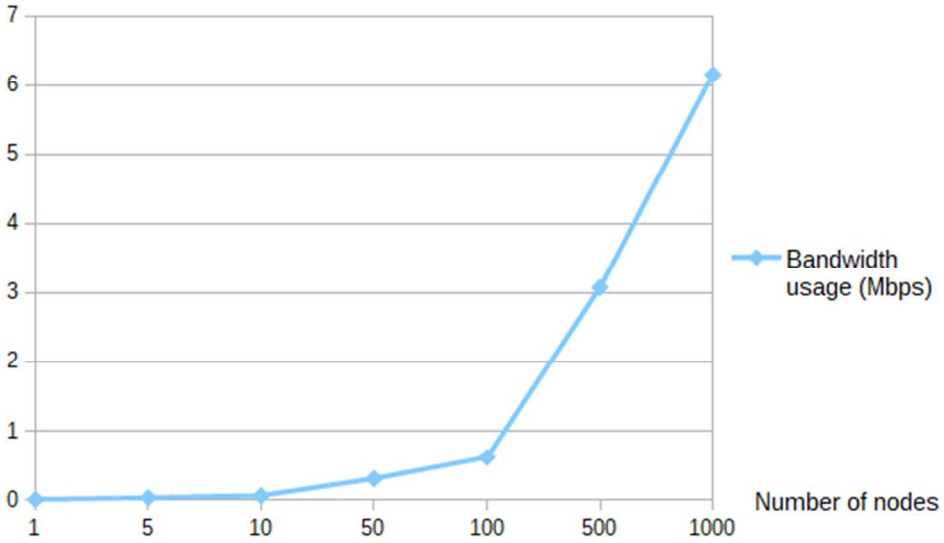
Detailed view of the incorrect normals generated.

Robotic arm in a dangerous pose after following an incorrect normal.



Monitoring the dangerous pose with the simulator in real-time.





Bandwidth usage of the local network stress test.