On-line simulation as a collision prevention layer in automated shoe sole adhesive spraying

Vicente Román-Ibáñez[1], Antonio Jimeno-Morenilla[1*], Francisco Pujol-López[1], Faustino Salas-Pérez[2]

[1]Computer Technology and Computation department
University of Alicante

[2]Spanish Footwear Technology Institute (INESCOP), Elda, Spain

*Data of the corresponding author:

Postal address:
Campus de San Vicente del Raspeig s/n
03001 Alicante, Spain

Phone: +34 96 590 34 00 Ext. 2453
Fax: +34 96 590 96 43
email: jimeno@dtic.ua.es

Abstract

Robotization in the footwear sector is a major challenge for the industry with difficulties present due to the inherent adaptive needs of some of the automated tasks performed by robotic arms. One of those of particular concern is collision prevention: working with those robots in automated on-line environments, considering they have limited awareness of the environment, may lead to dangerous situations with the movement of the robot along the calculated paths. To overcome this problem, a collision prevention layer based on a custom robotic software simulator is presented and justified in this paper to be used on automated shoe sole adhesive spraying cells. The performed experiments prove the feasibility of the proposed method in a real scenario with the speed and precision required by the automated task.

Keywords. Simulation, Robotics, Kinematics, Footwear, Collision prevention.

1 Introduction

Since the beginning of industrial automation, the main objective has been to increase the productivity of industrial production, reducing the time required for completion and increasing the economic profit of the result.

The footwear industry has been working mainly with handmade tasks in the production line from the beginnings until nowadays. There exist some reasons that explain the slow robotic adaptation of the tasks in the footwear industry [1], even with the advances in robotics in recent years.

- The flexibility required to successfully automate tasks. Such requirement is derived from the diversity of models, the different sizes of the same model with left and right feet, the quantity of different parts that make up the shoe and the big number of operations (up to 140, depending on the model) required to transform the materials into a finished product.
- Another important factor to take into account is the high precision required in some of the tasks in order to meet the industry requirements [2].
- Some of the tasks pose additional difficulties due to materials being non-rigid which makes it hard to grab and process them, or too slim / fragile to be treated for certain tasks such as roughing, noting that all these processes need to be performed better and faster than a human worker to be useful.

Being a burning issue, some authors proposed methods for several footwear tasks such as the application of adhesive in shoe sole gluing [3], roughing or grinding of the bottom surface of the shoe so that the gluing process works adequately [4], last milling to create shoe lasts based on custom measurements [5], polishing of the finished shoe [6] and even the final product packaging process has been robotised [7], all of them presenting the already mentioned problems of the footwear manufacturing sector. In fact, the above-mentioned automated tasks are rarely implemented in real factories.

For the automation in the footwear industry, robotic arms are proposed, just as other industries like the automotive, food or metal processing do. The main reason for using them is the ability to perform different tasks with the same machine and to adapt the same task to different models without hardware modifications.

The robotic arms used are usually composed of heavy metal joints moving at high speed. This may harm humans and destroy or damage expensive cell components and even the robotic arm itself when a tool is attached to the end effector.

The effects of robot mass and velocity of several industrial robots in critical body parts are studied in [8]. This study shows that most of the analysed robots are capable of reach speeds of 2 m/s and apply forces from 1.2 kN onwards. This fact proves the need and importance of collision prevention when dealing with such potentially dangerous machinery to prevent damage to the cell components and processed materials.

Collisions are one of the main reasons that hinder automation in the footwear industry. When the precision and flexibility requirements of the footwear processes are high, collision prevention becomes a must. With the high process diversity and precision required for small components, any minor mistake can cause damage to the final product. Being a critical issue, collision avoidance with robotic arms have been studied by other authors [9], [10] to provide specific solutions adapted to their needs. There exist algorithms to allow computers to detect collisions between virtual entities by software [11].

The idea of the research introduced here arises from a real footwear industrial problem of collisions in automated robotic cells with on-line processes. The main objective of this paper is to investigate and develop a new collision prevention layer for these on-line cells with minimal costs and hardware modifications, using an automated cell for adhesive spraying as a test bench due to the on-line characteristics of the process. The paper aims to bring automation closer to footwear sector, which due to the problems above mentioned is still far from being fully automated. This is an important qualitative enhancement for the integration of automation in footwear sector.

This article is organized as follows: Section 2 presents a short state-of-the-art in the robotic collisions field and justifies the need for extra prevention measures for on-line automated footwear applications. Section 3 explains the work flow of the robotic cell used in the experiments and describes the new collision prevention layer added to it. Section 4 describes the experiments performed and their results, and finally Section 5 presents some conclusions.

2 Collision issues in robotics for footwear

When the tool path is calculated from offline data, it is also possible to pre-calculate and test some collision prevention measures to be applied to each future model. Some of the prevention measures that can be considered using offline processing are listed below:

- Crop box: Ignore points outside a bounded work cube.
- Detect noise: Warn against strong joint angle acceleration.
- Test reachability: Check if all points are reachable by the robotic arm.
- Collision test: Check all reachable points for collisions in static environments.

In particular cases in which the tool path is dependent on some input data taken during the process instead of working with offline data, it is very difficult to determine whether it can be safely performed or not, and therefore to avoid collision issues.

The main problem with detections before path execution by the robot is the limited environment awareness those robotic arms have, due to the lack of sensors and the fact that they are simple mechanical tools that blindly move their joint motors to position a tool on specific world coordinates and orientations, fed with the computed CAD path. This can be problematic in footwear factories, where the automation

process is usually not complete and only a few tasks are automated, so workers need to interact with the cell controller and handle the input / output of materials.

There exist models that use computer vision systems to detect and avoid collisions inside the cell [12], including some of the well-known inexpensive depth sensors [13] such as Kinect® or ASUS Xtion®. Even if they are low-cost sensors, they add extra hardware and complexity to the cell that should be avoided.

Another drawback of the vision system comes with the occlusions that may occur with the moving parts of the robotic arm hiding lower parts of the cell and being unable to detect collisions in those situations. Also, when case a collision is detected, the robot may have already spent some time executing the tool path until the detection is made, probably making the piece that is being processed in the cell useless. For example, if the cell is roughing a shoe or gluing a sole and the collision is detected in the middle of the process, then the piece cannot be processed again from the beginning of the path. This problem can be solved by previously checking all the points in the path in a simulated environment. However, computer vision systems can be used together with other methods to add additional prevention layers where non-simulated entities interact with the cell.

An example of a potential hazardous scenario can be found in the process of applying adhesive to shoe soles using robotic arms. This is just an example to illustrate a real problematic scenario, but the proposed method in this paper is intended to be a general approach and not a single problem solver. In this process it is first required to obtain the path using a laser-scanned point cloud to be adapted to different sole sizes, models and placing positions. Figure 1 shows the point cloud obtained from the laser scanner, while in Figure 2 the mesh and path calculated with the CAD software from those points are shown.

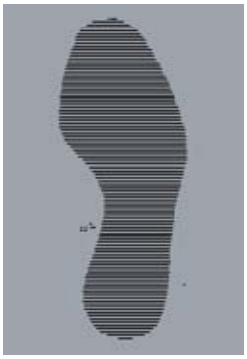

Figure 1:Scanned shoe sole point cloud.

Figure 2: Shoe sole mesh and path.

With every shoe sole scan being different to each other and affected by external noise such as light, occlusions or shiny surfaces, the quality and feasibility of the calculated path can be compromised, which may lead to uncontrolled movements of the robot joints with the associated danger. To show the problem, notice the difference between a normal shoe sole scanned mesh and the blue line path as shown in Figure 3 and another affected by direct sunlight during the scanning process as can be seen in Figure 4.

Noise leads CAD software to an incorrect mesh and path generation, and to obtain dangerous normals. Those normals appear in the zoomed version of the noisy scan in Figure 5 as red lines. With irregular and almost parallel angles to the sole plane, normals may lead to uncontrolled movements and collisions with the environment. These problems can be reduced by covering the 3D scanner with a controlled light tunnel, but it is not possible to be certain that paths are completely free of dangerous robot poses. Therefore, the proposed model is still needed to avoid possible collisions of the robot with other elements.

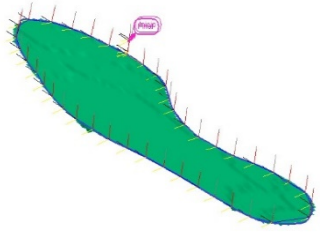

Figure 3: Scanned shoe sole.

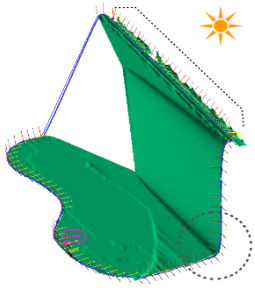

Figure 4: Incident light during

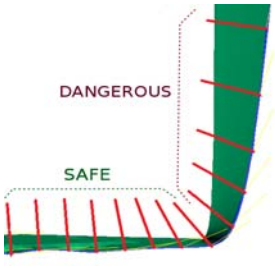

Figure 5: Detail of incorrect

scanning process. normal.

Even if the path is fully accurate, collisions with the environment may occur, for example, when some of the joint angles move the robotic arm to a position and orientation where the tool is reachable and free of collisions but some of the joints do not. This happens because usually only the tool position and orientation for each point in the path are taken into account when doing safety checks. However, some external methods can be used to give environment awareness to the robot for collision avoidance, such as computer vision [14].

Collisions of robotic arms with the environment can be potentially dangerous because these robots are composed of heavy metal joints moving fast. In addition, there is no direct human control of the robotic arm movements when in automatic mode, which is the default working mode in production environments.

In Figure 6 the resulting pose of the robotic arm for a point in the noisy path previously shown in Figure 4 can be seen, showing the possibility of a collision with the environment, in this case with the conveyor belt. The problem comes when the arm tries to reach the desired orientation of the noisy normal, with the aggravating circumstance that the twist between the previous normal and the current one is big. The result is a fast acceleration and movement of the joints, making it difficult to manually stop the robot arm in time.

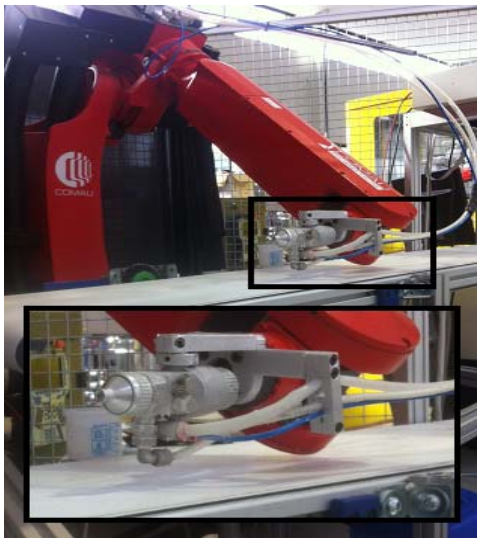

Figure 6: Robotic arm position after moving to a noisy normal.

3 On-line simulation layer for collision check

The usage of simulation in general robotic applications is not new and has been addressed by other authors [15]–[17]. However, due to the need of flexibility required by some of the footwear applications, the use of offline robot programming is not adequate or even possible, depending on the task to be performed.

Some of the advantages achieved by placing a simulator layer between the CAD software that generates the paths and a physical robot, constantly checking the validity of generated trajectories, are listed below:

- Enhanced collision prevention in on-line footwear cells with robotic arms.
- Reduction in the work space needed for the cell: less physical space required because the crop-box can be smaller due to constant collision checking of the simulator.

- Remote cell monitoring and control with low bandwidth usage [18].

In addition, there are other inherent advantages of the simulator for off-line environments that apply here such as:

- Design of the working cell before purchasing expensive equipment.
- Reduction in the cell cost, by choosing a cheaper/smaller robot that is able to do the same task in a specific environment.
- Reduction in the production stop time of hardware upgrades as they can be previously tested in the simulator.

Our proposed system is composed of all the different parts of the robotic cell and the extra layer with the simulator between the CAD software and the physical robotic arm, as can be seen in Figure 7. The model is intended to be used for a wide range of footwear robotic tasks, where an input device, a CAD software and a robotic arm should be present.

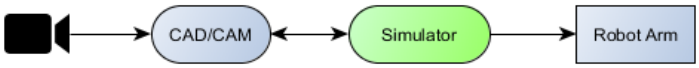

Figure 7: Schematic with the layers of the method.

The input device obtains the image feed that is later converted into a point cloud composed of depth data X, Y, Z. As an example, a laser scanner can be used to triangulate the laser position and obtain depth data grouped into sections [19], the result of which is shown in Figure 1.

That data is then transferred to the CAD software [20] in order to be filtered and to compute a path from the points [21], [22]. Figure 2 shows a generated mesh with path points for the previous example of a shoe sole. The generated points are then simulated in background using the added collision prevention layer to make sure they are accurate, reachable and safe. The CAD software and the simulator are connected through a TCP socket connection. Finally, the simulator uploads the checked path to the FTP server of the physical robotic arm to execute it and perform the desired task. If a collision or a non-reachable point is detected during the simulation, the full path is not transferred to the robotic arm to avoid collision problems. The rejected path is then stored and marked as not executed so it can be retried later or checked visually in the simulator. While the robot is executing the path, the scanner is processing the 3D data of the next sole to optimize the process. Is important to note that the reason not to use a real-time approach in this model is forced by footwear tasks itself. If real-time processing is used to simulate the path point by point instead of batching them and some problem is detected during the check, then the processed shoe part may end up unusable. The flowchart of the proposed method is shown in Figure 8.

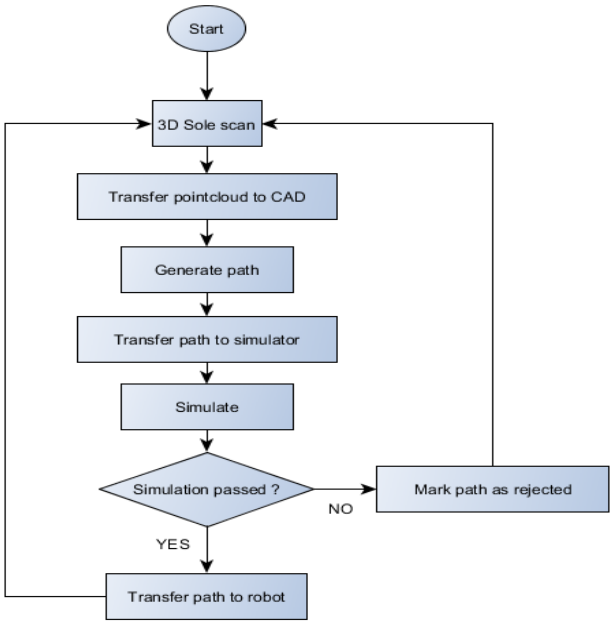

Figure 8: Flowchart of the proposed method.

The simulator makes use of kinematics to simulate the paths versus time omitting the forces originating them as it is performed in dynamics. Both forward and inverse kinematics are used to perform the simulation.

Forward kinematics allows the simulator to know the world position and orientation of each joint when rotations are applied to them locally. The standard Denavit-Hartenberg convention [23] has been used to describe the kinematic chain of the robotic arm in the simulator with the parameters given by the manufacturer and shown in Table 1.

Table 1: DH Parameters for Comau SmartSix

#	Θ	d	a	α
1	0	0.45	0.15	-π / 2
2	0	0.00	0.59	0.00
3	0	0.00	0.13	-π / 2
4	0	0.64707	0.00	π / 2
5	0	0.00	0.00	-π / 2
6	0	0.095	0.00	0.00

Each line of the DH table corresponds to a robot joint defined by only four parameters Θ, d, a and α. The parameter Θ represents the joint angle while d the joint offset. The link length is defined by a and the twist angle by α. With these parameters it is possible to obtain a homogeneous transformation matrix with equation (1). As a result of applying translations and rotations in the specified standard DH order, the resulting matrix T is capable of converting from coordinate system i^{-1} into i.

$$ {}^{i-1}T = \begin{bmatrix} cos\theta_i & -cos\ \alpha_i\ sin\theta_i & sin\ \alpha_i\ sin\theta_i & a_i cos\theta_i \\ sin\theta_i & cos\ \alpha_i\ cos\theta_i & -sin\ \alpha_i\ cos\theta_i & a_i sin\theta_i \\ 0 & sin\ \alpha_i & cos\ \alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad 1) $$

The successive application of those matrices shown in equation (2) allows the transformation between a pair of arbitrary joint's coordinate systems.

$$ T = \prod_{i=1}^{n} T_i \quad 2) $$

An example for a standard 6-DoF robotic arm using this transformation is defined in equation (3).

$$ T = {}^{0}T_6 = {}^{0}T_1\,{}^{1}T_2\,{}^{2}T_3\,{}^{3}T_4\,{}^{4}T_5\,{}^{5}T_6 \quad 3) $$

To that chain, the offset of the base T_{base} containing the translation and rotation offset from the ground to the first robot joint can be added. The same process is needed between tool offset and the end effector with T_{tool} leaving the forward kinematic equation (4).

$$ T_{final} = {}^{base}T_{tool} = T_{base}\,{}^{0}T_6 T_{tool} \quad 4) $$

Furthermore, inverse kinematics [24] are used to determine the joint angles needed to reach a specific position and orientation with a tool attached to the end effector of the simulated robotic arm based on the forward kinematics obtained from DH. Inverse kinematics is applied for each one of the points contained in the tool path generated by the CAD software. The input vector defined in equation (5) should contain both position p and orientation Θ of the desired path point to be followed by the end effector e of the robotic arm.

$$ e = [p_x, p_y, p_z, \theta_x, \theta_y, \theta_z]^T \quad 5) $$

After applying inverse kinematics, the resulting solution vector Θ contains the angle values of each robot joint that can be used to render the model inside the simulator. To this end, forward kinematics needs to be used, which in the 6 DoF arm used in the experiments has the form shown in equation (6).

$$ \theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6]^T \quad 6) $$

Since one of the goals of the current research is to prevent collisions of the robotic arm with the surrounding environment, a collision detection system module has been added to the simulator.

The collision system is based on the open-source project Open Dynamics Engine (ODE) [25] and is responsible for performing robot against environment collision checking using a hierarchical detection method [26]. Being open-source and a readily available library, it reduces the development time and cost of the proposed model. Experiments in the next section prove that method used in this library can run complete simulations in modest computers within the timeframe of online processes found in the footwear sector. If reader needs are not fulfilled with this collision detection method, it can be exchanged with one of other existing rigid methods [27]–[29].

The method in ODE first uses a fast check with AABBs (Axis-Aligned minimum Bounding Boxes) to locate regions of the space where a collision may have occurred, avoiding expensive checks in regions where collisions cannot exist in the current simulation step. Then, both the parts of the robot and the environment where AABBs were triggered are surrounded by OBBs (Oriented Bounding Boxes) [30] along the centroid of the objects. Figure 9 shows this approach, with two screws surrounded by both an ABB (blue boxes) and an OBB (green boxes). The collision boxes get the same rigid transformations as the parent object and allow the system to perform the collision tests between simpler shapes and therefore with less computational requirements, which is a must when dealing with on-line simulations. If more precision is required, more subdivisions of the collision subspaces of the hierarchy tree can be added. Figure 10 shows two OBBs colliding, the tool of the robot coloured in red and the robot controller.

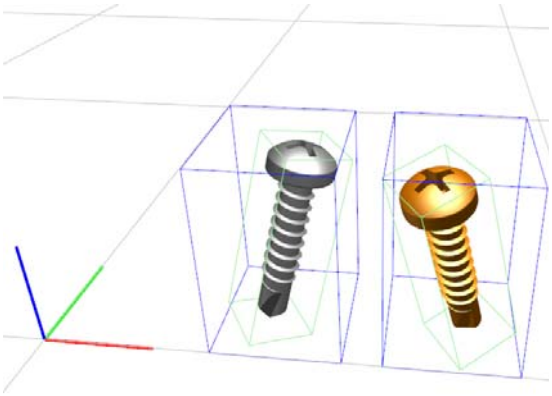

Figure 9: AABBs (blue) and OBBs (green).

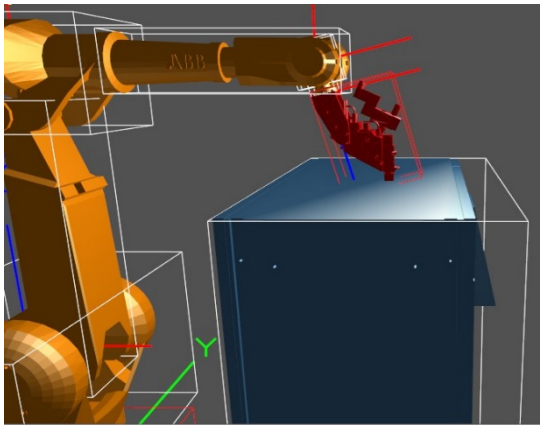

Figure 10: Oriented Bounding Boxes colliding.

Although the simulation is performed in background while the system is scanning the next sole, a GUI has been created with 3D render capabilities. The GUI shown in Figure 11 allows the user to define and manage the simulated environment and to perform a visual analysis of the generated paths, which is useful to know at which point they failed or to test the system before it is physically constructed.

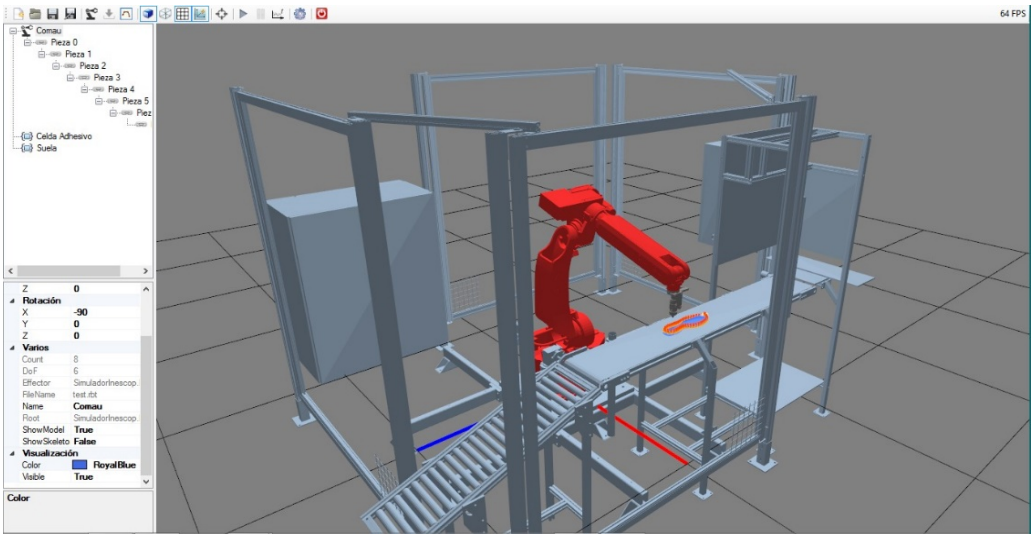

Figure 11: Simulation of the shoe sole adhesive spraying cell.

4 Experiments

In order to confirm the feasibility of the described method, some experiments were performed using an existing robotic cell for shoe sole adhesive spraying, which is shown in Figure 12. The reason to select this cell for collision prevention experiments is that it uses on-line path generation and it contains other hardware elements that can be reach by the robotic arm.

4.1 The sole gluing process

The cell is designed to perform the gluing process on shoe soles that it is normally realized by hand using brushes. This process is needed in the footwear industry to join the sole to the rest of the shoe. This task is especially important for the quality of the final product. Any problem associated with this process may lead to premature defects, reducing the reliability and the lifespan of the shoe. Having this task automated leads to a constant and predictable production rate and material costs, also ensuring the high precision needed in this industry.

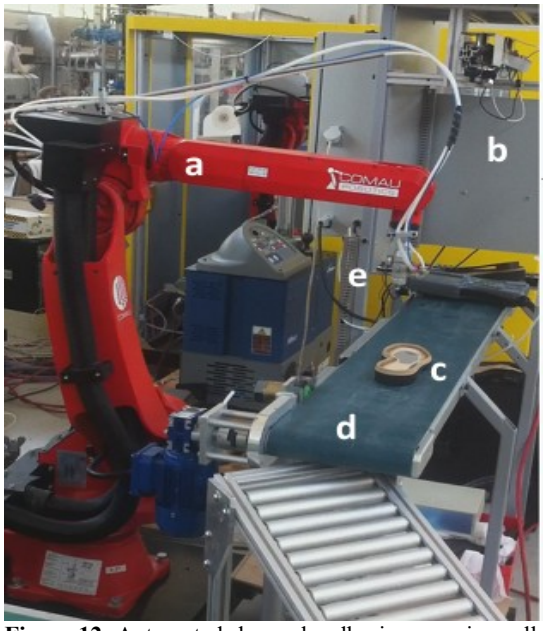

Figure 12: Automated shoe sole adhesive spraying cell. (a) Robotic arm. (b) Laser scanner. (c) Shoe sole. (d) Conveyor belt. (e) Adhesive spraying tool.

Figure 13: Shoe sole with adhesive applied.

4.2 Equipment

The robotic arm used in the experiments is a Comau SmartSix with 6 DoF with an adhesive spraying tool attached to the end effector and a pump to feed it. The triangulation-based laser scanner is custom made with a CCD RGB camera and a blue line laser module. The laser has a wavelength of 405 nm, in the range of blue colour, and a power of 20 mW. The camera has a resolution of 640 x 480 pixels with a polarized filter attached to it in order to capture only the wavelength of the laser. Also, a conveyor belt was used with an encoder to determine the offset position relative to the start of the movements. All these parts were connected to a control panel box along with other needed electronic components, while both the CAD and simulator software were contained in a computer with CPU i3-4150@3.5 GHz, 4 GB RAM and an integrated HD Graphics 4400 GPU.

The CAD/CAM software used to calculate the tool path from the point cloud obtained with the laser method is BasicCAM. The software has a socket waiting for TCP connections in background containing the scanned point cloud. The CAD software returns the list of points to be checked by the simulator and executed by the robotic arm inside the cell if the simulation does not return any problem. Figure 13 shows the resulting shoe sole after adhesive application by the spraying tool attached to the robotic arm.

4.3 Modelling scene

Finally, the cell was modelled inside the simulator and run in offline mode as shown in Figure 10. This made it possible to check if the robotic arm model was capable of performing the desired tasks in the current distribution of the cell elements.

Once the offline design process was finished, the collision prevention layer was tested through several experiments to check if it was working as expected and within an acceptable time range.

4.4 First experiment: Simulation time

The first experiment performed measured the total spent simulation time for a different number of tool path points but using the same scene objects. The simulated scene was the real shoe sole adhesive cell which contained 19 collidable parts divided into 12 for the environment and 7 for the robotic arm and the tool. The experiment took into account the loading time of the scene and also the simulation of all the points of the path. The path was generated by modifying a length parameter in the CAD software that altered the quantity of points of the path generated.

Figure 14 shows that the time spent by the model to perform the full simulation was suited to work with footwear applications, where the number of needed points in the path is small. In the current shoe sole application cell, the quantity of points used to describe the contour of the sole for the gluing task remained between 50 and 100. This small quantity of points led to complete simulations in less than a quarter of second. In addition, the graph shows that the simulator performed well with the increase of path points, having a constant loading time near to 220 ms at the start and a linear growth of the slope with the number of points. The loading time depends on the number of polygons loaded by the scene and is constant in this experiment because the same scene is used.

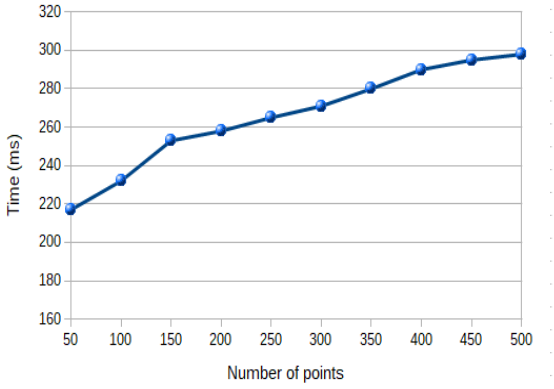

Figure 14: Graph showing the number of points vs time.

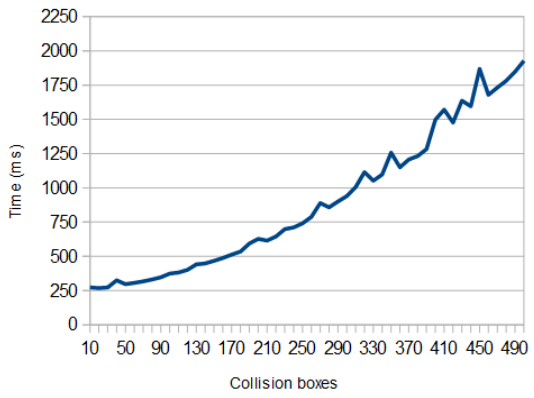

Figure 15: Graph showing the number of collision boxes vs time.

4.5 Second experiment: Collision testing

It is important to ensure that collision checking does not create a bottleneck in the simulation. To this end, an experiment was performed to compare the time spent with a different range of collision boxes.

The dataset for the experiment was created by generating a set of simulator scene files containing an incremental number of simple collision cubes per file. Rotation, scale and position of each object was set randomly, but in all the scenes the same number of points in the tool-path was used. Thus, only the collision-related process was tested.

Moreover, all the points of the path were simulated to test the worst-case scenario where the collision occurred in the last point. Note that in real world applications the simulation can be stopped on detecting the first collision to speed up, as a single collision is sufficient reason to avoid the task execution in the physical robotic arm to prevent damage to the cell components.

The resulting graph can be seen in Figure 15 where the start of an exponential progression was detected. However, an acceptable on-line performance was achieved, capable of executing the full simulation in less than a second, with up to 300 collision boxes. In comparison, the real shoe sole gluing cell used in the other

experiments contained only 19 collidable parts, resulting in a computational cost of less than 300 ms in collision checking for the same situation.

Previous experiments ensured that the simulator was able to simulate more complex scenes than the required footwear tasks in on-line with reduced overhead to the entire cell process.

4.6 Third experiment: Simulation accuracy

Once the computational and time requirements had been fulfilled, it was necessary to ensure the method's accuracy in checking collisions. To this end, two groups of 200 different paths each were created. The first one contained a set of paths that did not generate any problem. The second set contained paths that led to collisions or unreachable points in the scene.

The data in both datasets was obtained from real shoe sole data coming from the laser scanner. This data was processed by the CAD software to filter it and obtain a path containing a set of points. Finally, simulator scene files were created including each path returned by the CAD application. In order to generate the second set, some errors were injected into the normals of the paths, so the files led to collisions in the real robotic cell. All the tested scenarios in this experiment were performed with the same shoe sole gluing cell scene used in the first experiment, in order to obtain meaningful data.

With both datasets prepared, the set with correct paths was tested to make sure it was not rejecting good paths. Same process was performed using the set with incorrect paths, to ensure that no false negatives appeared. The results of this experiment can be seen in Table 2 as a confusion matrix.

Table 2: Confusion Matrix.

		Predicted	
		Good	Bad
Actual	Good	187	4
	Bad	13	196

The resulting confusion matrix displays a strong diagonal, meaning that the simulator is performing well, with a high ratio of true positives and negatives, as expected. The accuracy obtained is 0.9575. This value gives the proportion of correct guesses of the simulator.

On analysing the errors made by the simulator two different problem sources are identified. The first one is False Positives (FP), meaning good paths wrongly marked as dangerous. This is a minor problem compared to False Negatives (FN), because it blocks the execution of a good path but does not damage the environment. FP may occur because the area of the collision boxes used in the OBB detection method is always equal to or greater than the original mesh, sometimes triggering the collision before it actually happens. This can be reduced by changing the collision checking approach to another with greater accuracy such as the convex hull, taking into account the increase in time complexity. Also, it may be a good idea to have the error margin given by FP. That is because even if those paths could have been executed by the robot, they would have been very close to collide.

False negatives mean bad and dangerous paths passing through the simulator filter and executed in the physical robotic arm, although they were negligibly low in the current experiment. Those errors may appear due to an inconsistency between the real position and orientation of the physical cell parts and the simulated ones. The precision in the measurements between the real and the simulated world have a high impact on the simulation error output. Additionally, if a part of the cell must be moved or rotated, the simulation scene should be updated accordingly to avoid these kinds of problems. Another source of FN could be a small number of interpolation steps between the path points that could lead the simulator to jump over collisions during the simulation. This can be addressed by increasing the number of points interpolated in the simulated path, without compromising the on-line time constraint.

The results show that the simulator reduces drastically the collision problems present in these dynamic robotic cells.

With the data of the confusion matrix, a ROC graph has been generated as shown in Figure 16. The horizontal axis represents the False Positive Ratio (FPR) while the vertical axis shows the True Positive Ratio (TPR). The ROC curve generated plotting TPR against FPR illustrates the performance of the simulator. Every point above the diagonal means a performance higher than a random guess.

The resulting ROC curve is close to the perfect corner at (0,1), far away above the random guess line, with an area under the curve of 0.9575. This proves the consistent and reliable output of the simulations performed.

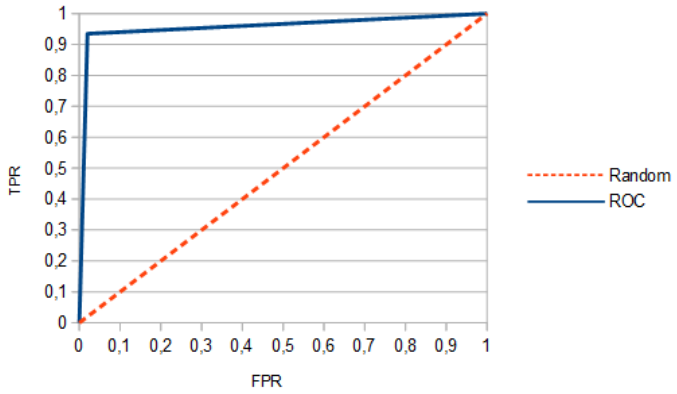

Figure 16: ROC graph.

5 Conclusion

The starting point of this research was the study of the difficulties present in the footwear industry when trying to automate tasks with robotic cells, mainly due to the flexibility and high precision needed to perform such tasks. In fact, there is plenty of papers about the topic, but barely few real footwear factories using them.

Then, due to the high variability and precision required in footwear automated tasks, collision issues arise especially when dealing with on-line environments. To overcome this problem, a model featuring a simulator layer has been designed and tested in order to reduce collision problems without compromising costs.

With this system, the paths automatically generated by the CAD software are verified to ensure their accuracy and to avoid collisions due to uncontrolled movements of the robotic arm inside the cell against other cell elements and the user work's area. The method has been implemented reusing the same simulator used to design the cell as an additional collision prevention layer with no extra hardware involved and working in on-line, without bottlenecking the automated process.

Although the current collision prevention model is not designed to control moving parts other than the robotic arm itself, it is more than enough to manage collision prevention in the described shoe sole robotic cell, where all the critical parts surrounding the arm are static.

Experiments showed that the method is not only feasible in terms of computational time requirements but also exhibits a high degree of reliability.

The proposed method is easily transposable to other industries with similar problems such as the furniture and toy industries.

Future works derived from this research will be aimed to increase the independence of the automation process for flexible related tasks in the footwear industry. This is intended to reduce the human interaction needed for such tasks, increasing the production and, hence, the profits.

6 References

[1] I. Maurtua, I. Goenaga, and A. Tellaeche, "Robotic Solutions for Footwear Industry," *IEEE*, pp. 2–5, 2012.

[2] M. Davia, A. Jimeno-Morenilla, and F. Salas, "Footwear bio-modelling: An industrial approach," *CAD Comput. Aided Des.*, vol. 45, no. 12, pp. 1575–1590, 2013.

[3] C. Wu, "Research on the generation of trajectory for shoe upper spraying based on structured light," in *2008 IEEE International Conference on Industrial Technology*, 2008, pp. 1–5.

[4] N. Pedrocchi, E. Villagrossi, C. Cenati, and L. M. Tosatti, "Design of fuzzy logic controller of industrial robot for roughing the uppers of fashion shoes," *Int. J. Adv. Manuf. Technol.*, vol. 77, no. 5–8, pp. 939–953, 2015.

[5] S. Xiong, J. Zhao, Z. Jiang, and M. Dong, "A computer-aided design system for foot-feature-based shoe last customization," *Int. J. Adv. Manuf. Technol.*, vol. 46, no. 1–4, pp. 11–19, 2010.

[6] L. Zlajpah and B. Nemec, "Robotic cell for custom finishing operations," *Int. J. Comput. Integr. Manuf.*, vol. 21, no. 1, pp. 33–42, 2008.

[7] R. Morales, F. Badesa, N. García-Aracil, R. Bormann, J. Fischer, and B. Graf, "Bimanual Robot Manipulation and Packaging of Shoes in Footwear Industry," in *ROBOT2013: First Iberian Robotics Conference*, vol. 252, Advances in Intelligent Systems and Computing, 2014, pp. 315–329.

[8] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger, "Safe Physical Human-Robot Interaction: Measurements, Analysis & New Insights," *Robot. Res.*, vol. 66, pp. 395–407, 2011.

[9] P. Chotiprayanakul, D. K. Liu, D. Wang, and G. Dissanayake, "A 3-Dimensional Force Field Method for Robot Collision Avoidance in Complex Environment," *ISARC Proc.*, 2007.

[10] T. D. Tang and E. L. J. Bohez, "A new collision avoidance strategy and its integration with collision detection for five-axis NC machining," *Int. J. Adv. Manuf. Technol.*, vol. 81, no. 5–8, pp. 1247–1258, 2015.

[11] R. Weller, "A Brief Overview of Collision Detection," in *New Geometric Data Structures for Collision Detection and Haptics*, no. September, 2013, pp. 9–46.

[12] D. M. Ebert and D. D. Henrich, "Safe human-robot-cooperation: image-based collision detection for industrial robots," in *IEEE/RSJ International Conference on Intelligent Robots and System*, 2002, vol. 2, no. October, pp. 1826–1831.

[13] P. Rakprayoon, M. Ruchanurucks, and A. Coundoul, "Kinect-based obstacle detection for manipulator," *2011 IEEE/SICE Int. Symp. Syst. Integr. SII 2011*, pp. 68–73, 2011.

[14] F. Flacco, T. Kroger, A. De Luca, and O. Khatib, "A depth space approach to human-robot collision avoidance," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 338–345.

[15] L. Zlajpah, "Simulation in robotics," *Math. Comput. Simul.*, vol. 79, no. 4, pp. 879–897, 2008.

[16] F. S. Cheng, "A methodology for developing robotic workcell simulation models," in *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, 2000, vol. 2, pp. 1265–1271.

[17] W. G. Hao, Y. Y. Leck, and L. C. Hun, "6-DOF PC-Based Robotic Arm (PC-ROBOARM) with efficient trajectory planning and speed control," in *2011 4th International Conference on Mechatronics: Integrated Engineering for Industrial and Societal Development, ICOM'11 - Conference Proceedings*, 2011.

[18] L. Wang, "Collaborative Robot Monitoring and Control for Enhanced Sustainability," *Int. J. Adv. Manuf. Technol.*, vol. (articule, pp. 1–3, 2013.

[19] Z. Hu, R. Bicker, P. Taylor, and C. Marshall, "Computer vision for shoe upper profile measurement via upper and sole conformal matching," *Opt. Lasers Eng.*, vol. 45, pp. 183–190, 2007.

[20] INESCOP, "Basic CAM: CAD/CAM solution for design shoe sector." INESCOP, 2011.

[21] Chuanyu Wu, Leiying He, Qinchuan Li, and Xudong Hu, "Research on the generation of trajectory for shoe upper spraying based on structured light," in *2008 IEEE International Conference on Industrial Technology*, 2008, pp. 1–5.

[22] V. Morell-Giménez, A. Jimeno-Morenilla, and J. García-Rodríguez, "Efficient tool path computation using multi-core GPUs," *Comput. Ind.*, vol. 64, no. 1, pp. 50–56, 2013.

[23] L. Radavelli, R. Simoni, E. De Pieri, and D. Martins, "A Comparative Study of the Kinematics of Robots Manipulators by Denavit-Hartenberg and Dual Quaternion," *Mecánica Comput. Multi-Body …*, vol. XXXI, pp. 13–16, 2012.

[24] S. R. S. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *Univ. California, San Diego, Typeset Manuscr. …*, vol. 132, no. 4, pp. 1–19, 2004.

[25] R. Smith, "Open Dynamics Engine ODE. Multibody Dynamics Simulation Software." 2004.

[26] S. Gottschalk, M. C. Lin, D. Manocha, and C. Hill, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *SIGGRAPH '96 Proc. 23rd Annu. Conf. Comput. Graph. Interact. Tech.*, pp. 171–180, 1996.

[27] M. Reggiani, M. Mazzoli, and S. Caselli, "An experimental evaluation of collision detection packages for robot motion planning," in *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, 2002, vol. 3, pp. 2329–2334 vol.3.

[28] C. Fares and Y. Hamam, "Collision Detection for Rigid Bodies: A State of the Art Review," *Int. Conf. Graph.*, 2005.

[29] T. D. Tang, "Algorithms for collision detection and avoidance for five-axis NC machining: A state of the art review," *CAD Computer Aided Design*, vol. 51. pp. 1–17, 2014.

[30] M. Lin, D. Manocha, J. Cohen, and S. Gottschalk, "Collision detection: Algorithms and applications," *Algorithms Robot. Motion Manip. (Proc. 1996 Work. Algorithmic Found. Robot.*, pp. 129–142, 1996.