Application for automatic programming of palletizing robots

Conference Paper · April 2018

DOI: 10.1109/ICARSC.2018.8374159

CITATIONS
5

READS
2,206

2 authors:

Frederico M. Moura
Instituto Superior de Engenharia do Porto
3 PUBLICATIONS **7** CITATIONS
SEE PROFILE

Manuel Silva
Instituto Superior de Engenharia do Porto
2 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Application for automatic programming of palletizing robots

Frederico M. Moura
Electrical Engineering Department, School of Engineering
Porto Polytechnic and INESC TEC
Porto, Portugal
frederico.matosmoura@gmail.com

Manuel F. Silva
Electrical Engineering Department, School of Engineering
Porto Polytechnic and INESC TEC
Porto, Portugal
mss@isep.ipp.pt

***Abstract* — Current market demands require several degrees of flexibility, speed, and repetitiveness of manufacture and logistic processes. Considering that a fourth industrial revolution is to be expected in a near future - which is highly based on smart machines, storage systems, and production facilities that cooperate to allow dynamic businesses and engineering processes - robotics presents itself as an increasingly sought-after solution, since it is often associated with such concepts. Hence, it is of no wonder that the worldwide operational stock of industrial robots has been increasing in a steady pace for the past decades and is expected to progress in such a trend. Within the several activities for robots on industrial applications, handling operations are regarded as predominant on the European market. Subsequently, palletizing applications are amongst the handling operations that have played an important role in the end stages of modern supply chains. In this context, this work aims to contextualise and develop an application for palletizing robots. This application, together with an off-line programming software (RobotStudio), allows for automatic programming of a robot's palletizing functions. Developed in the robot's native language (RAPID), the application has a basic user interface written in XML and can provide different pallet patterns.**

Keywords—*palletizing, robotics, programming, ABB, RAPID*

I. INTRODUCTION

The continuous search for methods and mechanisms that facilitate repetitive, high volume, or difficult tasks drives us, inevitably, to automate. Presently, and in spite the vast amount of applications for every-day employment of automation as substitute for human labour, industrial applications take on the biggest share of usage [1]. Current evolution of market demands imposed by recent trends, such as globalization or e-commerce, are continuously amplifying the pressure placed on manufacturers, regarding the way products must be manufactured, transported, and delivered. Shorter cycle times and quantities, different shapes and sizes are some of the challenges presented to production, packaging, and storing equipment and processes. Robotics is often seen as a part of a solution meant to deal with such demands.

Industrial robots are often associated with flexibility, and agility [2]. Attributes that are highly valued for cost reductions and increases on productivity. These properties have facilitated the use of robots across the whole spectrum of organisations' internal logistics, from goods inwards to goods outwards. Within the scope of applications that robots have played in the past decades, handling functions are one of the predominant, and will increase its relevance in the next years [3].

This work focus on palletizing, an important constituent of handling functions since it connects production to transport. It is a method of stacking goods, using one or more patterns, to assist on goods' handling, loading, storage, and other logistics related activities. The developed application aims at facilitating the task of programming palletizing robots. For the user, a minimal interaction with the software is required, confined to the input of a set of initial values and, if necessary, simulations. The Automatic Robot Palletizer (ARP) application is developed in the ABB's robot controller language (RAPID) and can be used together with its simulation environment (RobotStudio).

II. EXISTING SYSTEMS

The robot's controller contains its operating system, coordinating and executing several of the robot's functions, including its kinematic model, control algorithm, and the compilation/interpretation of the user program. Just as the robot's remaining components, the controller has been improved throughout time, due to the technological advances in electronics and, consequently, in software. One of these progresses allowed the possibility to program an industrial robot without having it physically present, process known as off-line programming. Up to that point, it was necessary to stop the robot to reprogram it, a procedure known as online programming, [4]. The need to incorporate sensor information and complex tasks to robot programs led to the creation of dedicated programming languages. The possibility to simulate the working settings, the controller, and peripherals within offline programming environments has allowed users to test intricate programs in a more reliable manner.

A. Programming languages

Robot programming languages are mostly developed by the robots' manufacturers. These languages are highly affected by the robot´s hardware and software technology [5]. Most programming languages designate sets of instructions, which although different in syntax, have several common properties. They were designed to create data structures and algorithms

using functions and sub functions, centred on the robot's movements. Depending on the type of language, they might be executed without being converted (using an interpreter) or compiled and executed (by a processor).

B. *Offline programming environments*

Offline programming software comprises several functions related to the robot's programming. These include the possibility to develop, compile, and simulate code for both the robot and accessories. Moreover, graphical environments allow the user to have a better perception of the robot's behaviour to the written code. They also facilitate the interface to the user and provide a tool to develop or optimize robotic cell layouts. Like the programming languages, most offline programming software packages are developed by the robot manufacturers, to provide the user with the necessary tools to use their products. However, there are currently several manufacturers of generic offline programming software [6-8]. These applications have the advantage of being able to program and simulate robots from more than one specific manufacturer.

C. *Palletizing software*

Several manufacturers of robots include in their standard software packages add-ins for palletizing-specific operations [9,10]. The inputs of these packages are commonly numerous and include the size of the objects, the pallet, type of robot and its tool, the stack configuration, among other parameters. There are also manufacturers that produce generic versions of this type of software, which generally include the same inputs [11,12]. Regarding the outputs, these vary from code (directly interpreted) to data files that require further software to be comprehensible to the robot's controller.

III. SYSTEM ARCHITECTURE

The concept behind the ARP application is to provide a quick method to implement palletizing functions on a robot, with less steps and parameters, when compared to existing software. User interaction with the system has been limited to a bare minimum, meaning that the user is only responsible for the input of the application's initial values and for modelling the existing robotic cell to simulate and test the robot´s behaviour, if necessary. Fig. 1 depicts the system block diagram and its connections to external modules or programs. The ARP application takes a set of initial values as inputs, entered on a specific Extensible Markup Language (XML) file (ARPInit). These values are linked to several different variables in the program. The developed application is comprised of several functions (or procedures), and information is exchanged among these functions via function parameters or global variables.

The different functions take on the tasks necessary to perform both the control and supervision of the robot's movements, the calculation and creation of the pallet layout, as well as additional auxiliary functions. An interface to a Programmable Logic Controller (PLC) is also incorporated, to provide a mean of information exchange to the robotic cell controller. All the functions or procedures are divided among three modules (either program or system modules): ARPPalletizer (contains the main program execution and additional data), ARPMovement (contains the functions that deal with the robot's movements) and ARPBasics (contains several functions that, although necessary to the execution of the program, are not essential to the main function).

Alongside the provided application main modules, the ARPInit file is to be downloaded to the virtual robot controller for simulation. After testing the program in simulation, the program can be downloaded to the physical controller. However, the possibility to download the necessary files directly to a physical robot without having to simulate beforehand is also foreseen.

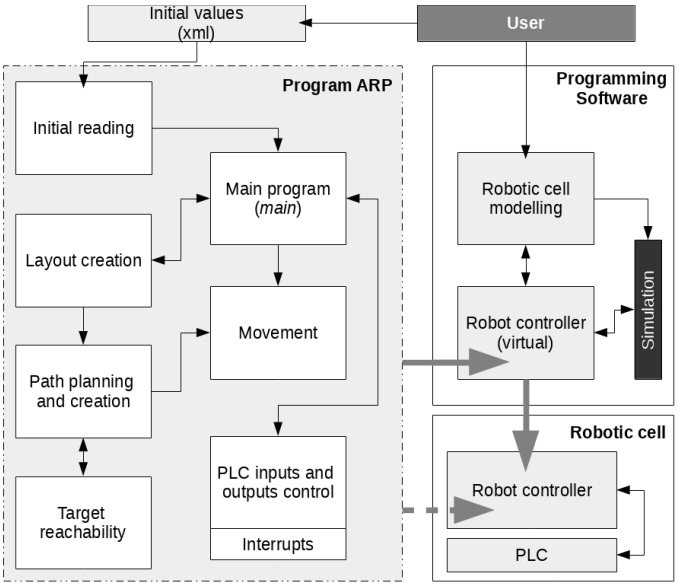

Fig. 1. System block diagram

IV. ARP STRUCTURE

A. *Program flow*

Fig. 3 depicts the minimal setup required for running the ARP application. This comprises the robot itself, a pickup location (or object feeder) for the robot (*i.e.* a roller conveyor), and a location where the stack will be built (*i.e.* a pallet). Two points are regarded for each location: an approach position (i.e. `px0`) and a final position (i.e. `px1`). Also on Fig. 3 is the robot's initial point (`pHome`).

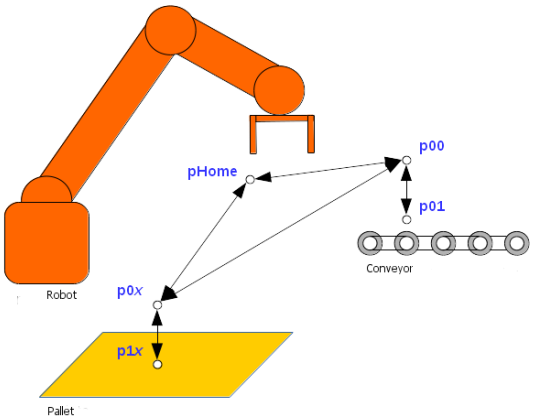

Fig. 2. System block diagram

The flowchart depicted in Fig. 3 summarizes the main program execution. At the beginning of the cycle the initial values are read from the XML file (using a parser written within the "ARPBasics" module), the corresponding variables in the program are initialised and the pallet layout calculated (as explained in Subsection IV.D.). Next the robot's tool position is verified: if it is not at the starting position, the tool is moved to `pHome`. The main cycle is then executed, starting by checking whether the program should terminate, on which case the robot is moved to its starting position. After, the presence of the pallet and the object are tested: if both are present, the robot's tool is moved to the pickup approach point (`p00`). This position, directly over the object, allows a less controlled and quicker movement of the robot. The final position, at which the object is picked up by the tool at a slower speed, is the final pick up point (`p01`). If there is an error during pick-up the cycle is aborted. The robot end-effector is then moved quickly to the delivery approach point (`px0`) and slowly lowered to the final position (`px1`), exiting the cycle in case of error.

B. Initial values

As a mean to provide a user interface, the initial values file (ARPInit) was designed to be readable by humans and machines, and to support a variety of platforms. For such a task, XML was chosen, providing a mean of composing a data structure. The data is distributed within three major categories:

- "*System*": contains general system values, such as the robot's starting position, its maximum reachable height and the tools' TCP value;
- "*Object*": values related to the position, orientation, and size of the pallet and of the objects that will be placed onto the pallet;
- "*Layout*": layout and layer values (such as layout orientation on both even and odd layers) that provide data to the controller to determine the objects' positions automatically or being read from manual input.

C. Targets and paths

In defining the pickup and delivery points, a WorkObject (*i.e.* a reference frame) was created per area and given a position and orientation related to the robot's *world* frame. These were named, respectively, *wobj_conveyor* and *wobj_pallet*. The pickup point is unique. In contrast, there are several delivery points. Furthermore, each point requires a specific orientation referenced to its corresponding WorkObject, which is given regarding the orientation of the layout (in the delivery area) or is kept at a constant value, as in the pickup area.

D. Pattern creation

The most common palletizing stacking styles are overlap and interlaced [13]. The overlap style is achieved by placing the objects with the same pattern in every layer, while the interlaced stacking style has a different pattern of even and odd layers. The developed ARP application can provide both styles. The choice is made by the operator in the ARPInit file.

Assuming a constant size of the object to be palletized, and considering the dimensions of both the object and of the area where the stack will be created, each object's individual position in the layout is calculated. All positions are related to a starting point, located on the origin of the correspondent WorkObject frame. To create a layer, first the limits are calculated per axis (number of items per axis) and then the offset of each object regarding the origin of the work object *wobj_pallet*.

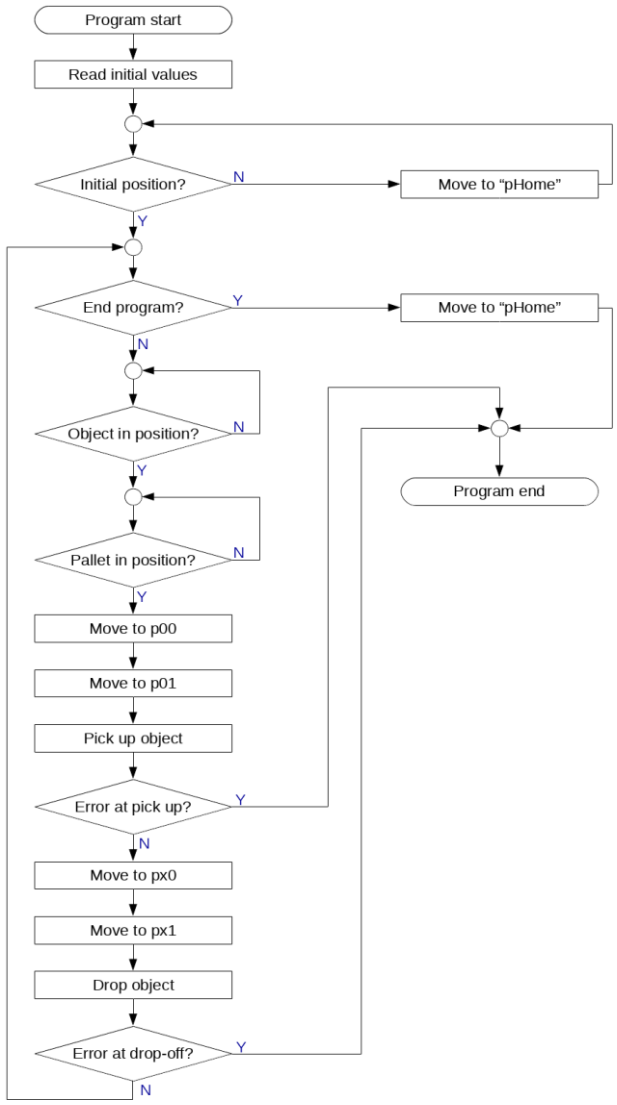

Fig. 3. Software flowchart depicting the main steps in the palletizing cycle

There are two types of distribution for the items: uniformed and non-uniformed. In the first case, the remaining space is divided in half and positioned on the extremities of the pallet. A non-uniformed distribution divides the space in equal amounts and distributes it between the objects and the extremities.

The object orientation is directly related with the layer orientation, which has four possible values: 0°, 90°, 180° and 270°. The orientation of the object at the pickup point is considered constant and of 0°. The remaining orientations are determined from the 0° orientation by rotating the object 90° around the *zz*' axis at the drop-off. The number of possible layers is obtained by subtracting the pallet height from the robot's maximum reachable height and the distance between

the approach and final points. This value is afterwards divided by the height of the object, truncating the result to obtain an integer number of layers [14].

E. PLC Interface

During the main cycle, information is being exchanged with the robotic cell controller (usually a PLC). This interface consists of a group of digital signals that were defined as a basic hand-shake between both controllers.

Table I presents the inputs and outputs present on the robot's side, along with the name of the variable to which they are connected internally on the ARP application.

TABLE I. PLC INTERFACE SIGNALS

Type	*Signal*	*Internal variable*	*Description*
Input	di1	diCycleStart	Palletizing cycle start
	di2	diCycleRestart	Palletizing cycle restart
	di3	diCycleStop	Palletizing cycle stop
	di4	diBoxReady	Box is ready at pickup
	di5	diPalletReady	Pallet is in place
	di6	diObjectSensor	Object is in tool
Output	do1	doRobotReady	Robot is ready to start
	do2	doVacuumOn	Turn vacuum on
	do3	doVacuumOff	Turn vacuum off
	do4	doRequestBox	Request a box
	do5	doRequestPallet	Request a pallet

V. SIMULATIONS AND TESTS

A. Methodology and parameters

To test the developed ARP application, ABB's RobotStudio, was used. This required the development of the virtual model of a robotic cell, including the definition and placement of all necessary equipment, and the logic which governs the cell's inputs and outputs.

The used robot is the ABB IRB460. As work tool, the vacuum gripper from ABB was selected. Shown on Fig. 4 is the placement of the robot and the delivery and pick-up WorkObjects. The developed modules were downloaded to the virtual controller and, using the available compiler, tested for errors. The first tests focused on the robot's movements, using two targets created to represent the pickup and delivery points. The program uses two types of movement: joint and linear movement. The first one is used to move the robot quickly in a path that is not required to be in a straight line. The specified velocity for these movements was 5000 mm/s (`v5000`) with a 15-mm corner path (`z15`). The motion from the approach to the final positions requires the TCP to slowly move in a vertical linear path, corresponding to a linear movement. The speed was lowered to 1000 mm/s (`v1000`) and the zone data was set to fine, to provide a more accurate positioning.

The initial tests made in RobotStudio were to determine whether the ARP program was correctly creating the layouts and stacks. After assessing that the program was providing the expected results, the program was further tested by verifying its response to variations to the initial values and to their corresponding internal variables (in the program).

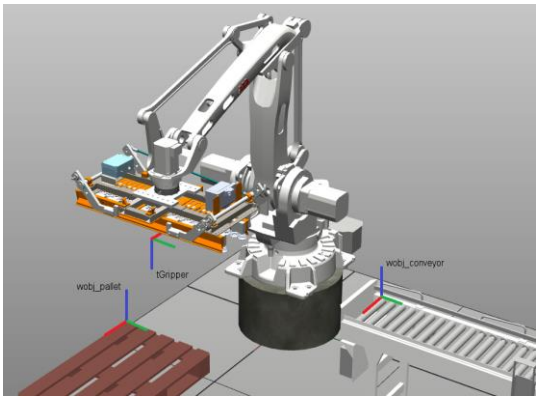

Fig. 4. WorkObjects of the pickup and delivery points, respectively *wobj_conveyor* and *wobj_pallet*, and the robot's TCP frame *tGripper*.

B. Tests in simulated environment

The simulations using the virtual robotic palletizing cell were divided into sets. In the first set the pallet dimensions (delivery area) were kept constant and the object's dimensions were varied. The second set of simulations had an object of fixed dimensions and a delivery area that changed size. The final simulations performed were of the real robotic cell available at the laboratory and followed the same test procedure as the previous sets. During each set, several simulations where made using different rotation angles, types of stacks and layout styles to aggregate as much diversity of stacking options as possible, within the working limits of the robot. Table II shows a group of eight cases that summarize the several tests that were made during the first set of the virtual robotic cell.

TABLE II. OBJECT SIZES OF THE FIRST SET OF CASES

Case	1	2	3	4	5	6	7	8
Length (mm)	200	150	225	400	400	750	500	800
Width (mm)	200	150	325	250	400	350	600	600
Height (mm)	125	400	200	300	400	550	700	600

All tests had the same pallet size of 1200 × 1000 × 144 mm (length × width × height), which is the standard size for an EPAL 2 Pallet [15]. Fig. 5 illustrates some of the virtual robotic cell simulations in the RobotStudio environment. These are of the third and sixth cases, respectively.

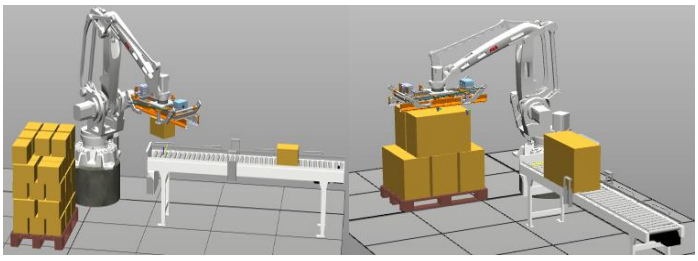

Fig. 5. Virtual robotic cell during the simulations of the third case (left) and of the sixth case (right), both with uniform distributions in interlaced style

A summary of the simulation results achieved for the first set is shown on Table III.

TABLE III. RESULTS OF THE FIRST SET OF CASES

Case	*Number of elements*		*No. of layers*	*Stack height (meters)*	*Duration (seconds)*	
	Overlap	*Interlaced*			*Overlap*	*Interlaced*
1	288	288	12	1,645	965	965
2	120	120	3	1,345	403	403
3	57	57	6	1,345	196	196
4	42	42	5	1,645	132	137
5	18	18	3	1,345	62	62
6	6	5	2	1,245	22	18
7	4	4	2	1,545	16	16
8	4	3	2	1,345	15	11

In the second set of simulations the object's size was kept constant and equal to 225 × 325 × 200 mm (length × width × height) and the delivery area was changed, as shown on Table IV.

TABLE IV. OBJECT SIZES OF THE SECOND SET OF CASES

Case	*Length (mm)*	*Width (mm)*	*Height (mm)*	*Type*
1	1200	800	0	unitizing
2	800	600	144	EPAL 6

The first case had the stack executed disregarding the height of a possible object between the floor and the stack. This operation is called "unitizing" and consists in the creation of a stack on top of a thin cardboard sheet (interlayer). In the second case an EPAL 6 type pallet was used (roughly half the size of the EPAL 2 pallet type) [15]. Examples of the simulations are shown in Fig. 6 and the summarized results of the set are displayed on Table V.

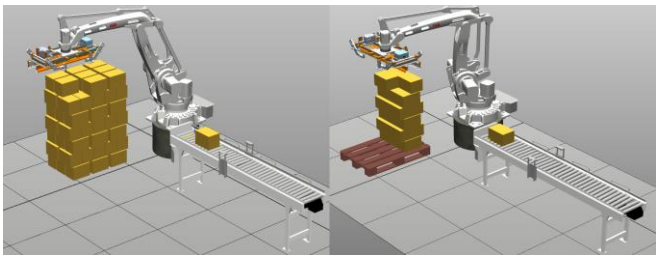

Fig. 6. Virtual robotic cell during the simulations of the second set, first case (left) and second case (right) using an interlaced stacking style for both cases with uniform and non-uniform distributions

TABLE V. RESULTS OF THE SECOND SET OF CASES

Case	*Number of elements*		*No. of layers*	*Stack height (meters)*	*Duration (seconds)*	
	Overlap	*Interlaced*			*Overlap*	*Interlaced*
1	66	66	7	1,600	226	226
2	21	21	6	1,344	74	74

C. Tests in physical environment

The aim of the final tests was to try out the ARP application to full extent under working conditions. These tests were centred on an ABB IRB140 robot (1), equipped with a Schunk PGN-64 two-finger parallel gripper (3), and respective IRC5 controller (2) (Fig. 7).

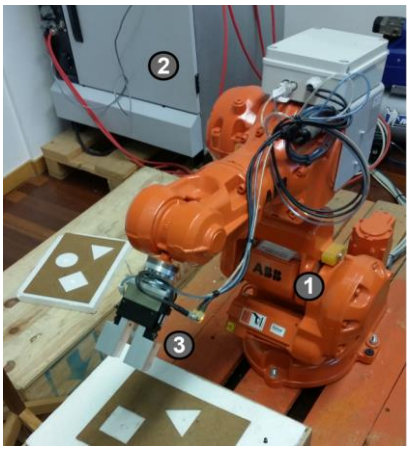

Fig. 7. IRB140 Robotic cell available on the laboratory

A simulation using the corresponding model of the robotic cell was made, prior to the laboratory tests, as per normal working sequence. The IRB140 is a six-axis robot with a small payload when compared with the four-axis robot (IRB460) chosen for the functional tests. As such, in the beginning of the simulations, the parameters on the ARPInit file had to be readjusted. Fig. 8 shows the virtual robotic cell, including the WorkObjects that ARP requires (*wobj_pallet* (1), *wobj_conveyor* (2)), and the TCP frame (3).

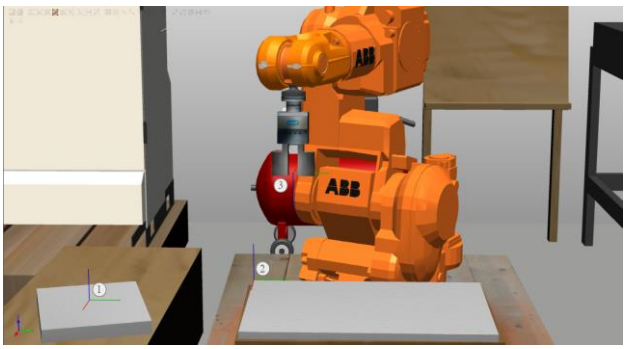

Fig. 8. Placement of the WorkObjects on the IRB140 robotic cell model in RobotStudio

The test structure for these simulations was the same as in the previous case. Due to the short motion range of the robot's gripper fingers, the size of the objects to be stacked was limited to a maximum of 20 mm, as shown on Table VI. This table displays the dimensions of the objects for both cases.

TABLE VI. OBJECT SIZES OF THE FIRST SET OF SIMULATIONS

Case	*Length (mm)*	*Width (mm)*	*Height (mm)*
1	16	16	11
2	20	40	10

The delivery area for both cases was of 50 × 50 mm (length × width), disregarding the height, since the stack was created directly on the surface on which the WorkObject was placed. Some examples of the created stacks are displayed on Fig. 9, showing an example for the first case on the left side and another example for the second case on the right.

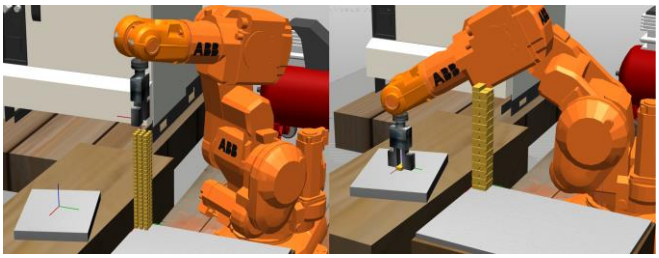

Fig. 9. IRB140 results during the simulations of the first case (left) and second case (right) of the first set, both with uniform distribution in an interlaced style

The left picture concerns a test that had an overlap stacking style with a non-uniform distribution and the right picture had also a non-uniform distribution but was executed with an interlaced style. The results of the two cases are displayed in Table VII.

TABLE VII. RESULTS OF THE FIRST SET OF SIMULATIONS

Case	*Number of elements*		*No. of layers*	*Stack height (mm)*	*Duration (seconds)*	
	Overlap	*Interlaced*			*Overlap*	*Interlaced*
1	234	234	26	286	421	421
2	58	58	29	290	115	115

The second set of simulations had delivery areas with the dimensions shown on Table VIII while the dimensions of the object were kept at 20 × 40 × 10 mm (length × width × height). As in the first set, and since there is no object on which the stack is to be built, the height was ignored.

TABLE VIII. DELIVERY AREA SIZES OF THE SECOND SET OF SIMULATIONS

Case	*Length (mm)*	*Width (mm)*	*Height (mm)*
1	40	80	0
2	120	100	0

Fig. 10 depicts the results of this set, with images of the simulation of both cases, and Table IX displays the achieved values.

TABLE IX. RESULTS OF THE SECOND SET OF SIMULATIONS

Case	*Number of elements*		*No. of layers*	*Stack height (mm)*	*Duration (seconds)*	
	Overlap	*Interlaced*			*Overlap*	*Interlaced*
1	116	116	29	290	210	210
2	435	393	29	290	792	712

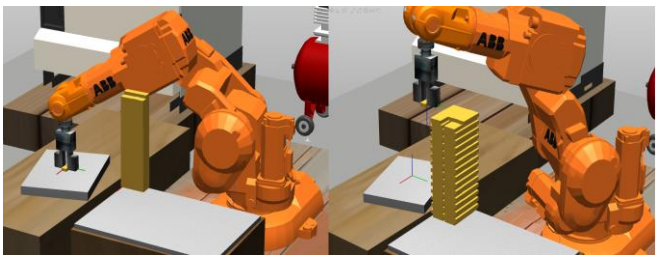

Fig. 10. IRB140 results during the simulations of the first case (left) and second case (right) of the second set, both with uniform distribution in an interlaced style

The values used on both sets of simulations were later used on the real robot controller to validate the simulations. The structure of the sets remained unaltered. The final tests began by downloading to the robot´s controller the same ARP files. The speed of the robot was initially reduced as a safety measure to avoid possible damage to robot. The results were similar to the ones obtained in simulation, with an increase in duration given the decrease in speed. Finally, setting the robot´s speed to its working value produced results equal to the ones achieved in the simulation environment.

VI. CONCLUSIONS

The ARP application described in this paper aimed at facilitating the programming of repetitive palletizing operations for robots using RobotStudio. The inputs and user interaction were kept at a bare minimum, simplifying the processes, during start-up or in case of rearrangement, reducing setup time. It also provides to the user a possibility to fully use the simulation functionalities provided by RobotStudio and, as such, avoid programming errors.

REFERENCES

[1] G. Bekey, J. Yuh, “Status of Robotics”, IEEE Robotics & Automation Magazine,2008

[2] J. Norberto Pires, “Industrial Robots Programming. Building applications for the factories of the future”, Springer, 2007

[3] Technavio, “European Industrial Robotics Market 2016-2020 | Market research reports,” [Online]. Available: https://www.technavio.com/report/europe-robotics-industrial-robotics-market. [Accessed February 2018].

[4] T. Lozano-Peréz, “Robot Programming“ in Proceedings of the IEEE, Vol.17, No. 7, July 1983

[5] S. Yang, X. Mao, B. Ge, S. Yang, “The Roadmap and Challenges of Robot Programming Languages” in IEEE International Conference on Systems, Man, and Cybernetics, 2015

[6] RoboDK, „Simulator for industrial robots and offline programming”, [Online] Available: https://robodk.com/. [Accessed February 2018].

[7] Visual Components, „3D manufacturing simulation and visualization software”, [Online] Available: https://www.visualcomponents.com. [Accessed February 2018].

[8] Octopuz, „Robot programming and simulation software”, [Online] Available: https://octopuz.com. [Accessed February 2018

[9] ABB, “Palletizing Software”, [Online] Available: http://new.abb.com/products/robotics/application-software/palletizing. [Accessed February 2018].

[10] KUKA, “KUKA.FlexPal”, [Online] Available: https://www.kuka.com/en-gb/products/robotics-systems/software/application-software/kuka_flexpal. [Accessed February 2018].

[11] Packer3D, “Packer3D”, [Online] Available: http://www.packer3d.com/. [Accessed February 2018].

[12] Koona, “Quick Pallet Maker”, [Online] Available: http://www.koona.com/qpm/. [Accessed February 2018].

[13] C. Hongtai, X. Yunjie, L. Yumei and H. Lina, “Design of Palletizing Algorithm Based on Palletizing Robot Workstation,” in Conference on Real-time Computing and Robotics, Angkor Wat, Cambodja, 2016.

[14] F.Moura, “Aplicação para programação automática de robôs de paletização”, Portugal, School of Engineering - Porto Polytechnic, 2017

[15] EPAL - European Pallet Association e.V., “Pallet Overview,” EPAL - European Pallet Association e.V., 2017. [Online]. Available: https://www.epal-pallets.org/eu-en/load-carriers/overview/. [Accessed February 2018].