AdaptPack Studio: Automatic Offline Robot Programming Framework for Factory Environments

André Castro, João Pedro Souza, Luís Rocha
INESC TEC
Porto, Portugal
{andre.l.castro, joao.p.souza, luis.f.rocha}@inesctec.pt

Manuel F. Silva
INESC TEC and ISEP-IPP
Porto, Portugal
mss@isep.ipp.pt

***Abstract*—The brisk and dynamic environment that factories are facing, both as an internal and an external level, requires a collection of handy tools to solve emerging issues in the industry 4.0 context. Part of the common challenges that appear are related to the increasing demand for high adaptability in the organizations' production lines. Mechanical processes are becoming faster and more adjustable to the production diversity in the Fast Moving Consumer Goods (FMCG). Concerning the previous characteristics, future factories can only remain competitive and profitable if they have the ability to quickly adapt all their production resources in response to inconstant market demands. Having previous concerns in focus, this paper presents a fast and adaptative framework for automated cells modeling, simulation and offline robot programming, focused on palletizing operations. Established as an add-on for the Visual Components (VC) 3D manufacturing simulation software, the proposed application allows performing fast layout modeling and automatic offline generation of robot programs. Furthermore, A* based algorithms are used for generating collision-free trajectories, discretized both in the robot joints space and in the Cartesian space. The software evaluation was tested inside the VC simulation world and in the real-world scenario. Results have shown to be concise and accurate, with minor displacement inaccuracies due to differences between the virtual model and the real world.**

***Index Terms*—Industry 4.0, Offline programming, Factory layout modeling, Product palletizing.**

I. Introduction

There is a current technological revolution changing the production and consumption of goods and services. The alleged Fourth Industrial Revolution has become marked by rising innovation advancements in various fields, including robotics and artificial intelligence. The technological processes are becoming increasingly connected and intelligent. As a result, production systems that tend to emerge in the future should have abilities to operate in real time, with instantaneous data acquisition and processing, allowing real-time decision making. These systems are becoming more virtualized, with copies of the original factories, allowing remote traceability and monitoring of all processes through the various sensors spread throughout the plant. Moreover, decentralization and modularity will act with decision making, made by service-oriented software architectures, according to the needs of the production in real time. Products will be manufactured according to demand, using adaptative manufacturing cells, which offers flexibility to modify machine tasks quickly.

Regarding palletizing cells, products life-cycle are becoming shorter, and some production lines are being converted to modular production cells [1]. On top of that, industries can only remain competitive and profitable in case they conduct flexible responses to the inconstant market, producing a wide assortment of items, as indicated by the client's determinations [2]. Long programming time of industrial robots significantly lowers production flexibility and volume. In consequence, it is crucial to have tools that allow the rapid programming of those robots. To accomplish the previous requirements, using offline programming is the usual approach. This type of programming relies on a graphical simulation of the environment, and the primary idea is to use computer graphics to create a virtual representation of the working cell. Inside this abstraction it is possible to manipulate all tasks and variables, conforming to the real situation.

Robot-based palletizing cells have captured attention in the last decade in the scientific sphere. Yu *et al.* [3] developed an offline tool for stacking objects using a MOTOMAN-HP20 robot and the brand's simulator [4]. On its turn, Kito *et al.* [5] produced an offline teaching tool for robots to pick bolts from a pallet. However, there is no automation in the previous solutions: the user still programs the robot manually, and high efficiency is still not achieved. Following a different approach, the work presented in [6] presents an autonomous smart robot with advanced sensing and decision-making capabilities to teach "child" robots in the production line. This last work, however, does not deal with the palletizing problem. For the specific problem addressed in this paper, Moura *et al.* [7] achieved automatic palletizing solutions for ABB robots. Despite the focus of previous solutions on a specific robot brand and its native simulation software, there are several manufacturers of generic offline programming software [8]–[10], which can perform programming of global robotic platforms. There are also alternative simulation software packages in the market [11]–[13], but they do not perform detailed simulation of robot motions and offline robot programming. The work

This work is co-financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 under the PORTUGAL 2020 Partnership Agreement, and through the Portuguese National Innovation Agency (ANI) as a part of project AdaptPack: POCI-01-0247-FEDER-017887.

978-1-7281-3558-8/19/$31.00 ©2019 IEEE

proposed by Silva *et al.* [14] generated a solution for the automatic offline generation of collision-free robot programs. In their work, a framework which aids the programming and path generation of palletizing routines has been developed. This platform has its base on the Visual Components (VC) simulation software [8], due to its capacity to perform detailed simulations and offline programming for several distinct robot brands over other candidates.

Considering the past ideas in mind, the present paper focuses on the development of a software framework responsible for facilitating the design, development, simulation, and effective installation of robotic palletizing cells. The AdaptPack Studio is a modular system designed as an add-on to the VC simulation software. This add-on engages the presented issue through three modules: palletizing cell design, modeling, and simulation; automatic offline palletizing routine generation; translation and exporting of generated programs to real systems. For this reason, it was constructed three specific systems inside the main platform to handle each module individually. Those modules are connected and exchange data among themselves, as illustrated in Figure 1.

The AdaptPack project main objective is to develop a framework, based on concepts of modularity, to accelerate the programming of robots devoted to palletizing tasks. To address this problem, the following project objectives were specified:

- Develop a solution for the design, and simulation of robotic cells, based on a library of existing components, allowing the assembly of palletizing systems in a short time.
- Develop a solution to generate automatic offline routines for palletizing products with different configurations, formats, and dimensions.
- Develop a robot path planning solution with collision avoidance.

This paper is organized as follows. Section II indicates the general concepts behind this project. After, Section III introduces the central architecture of the proposed system. Section IV presents the performed tests and results. Finally, in Section V, the main conclusions of this work are exposed, followed by a few ideas concerning possible future improvements.

II. Proposed Solution

One of the first steps when creating a new palletizing solution, is the design and modeling of the cell, either on paper or through virtual systems. Using computer graphics to model this solution from scratch regularly requires some effort to identify, choose and model the desired components. To expedite the modeling process, along the project were developed the models of general components used in palletizing operations. Those components were added to a library inside VC (named ”AdaptPack Building Blocks”), to form the ”Design and Modeling Module”. This module allows users to drag and drop components inside the cell model layout, and connect them as blocks.

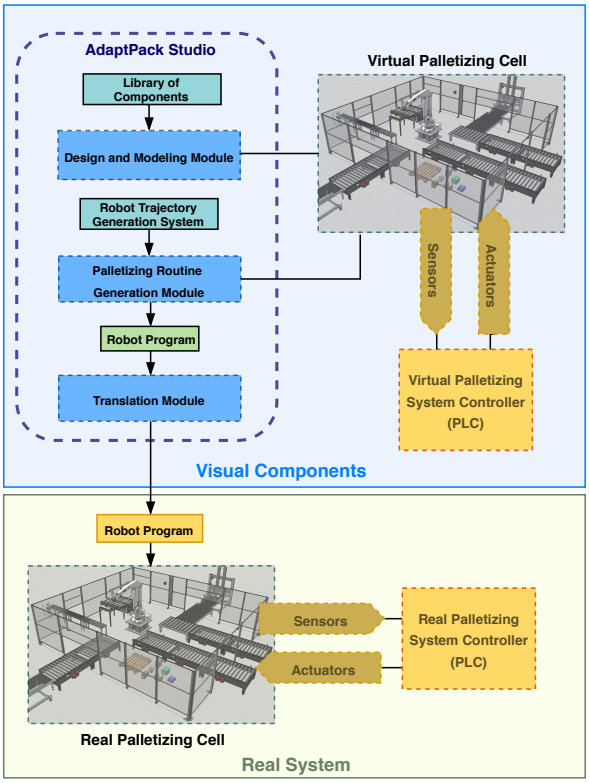


Fig. 1. The general block diagram of the system architecture, including the AdaptPack Studio and its main modules.

After modeling the cell, users can interact with a Graphical User Interface (GUI) to perform the offline programming of the robots. This GUI was developed using the C# Application Programming Interface (API) provided by VC, granting the possibility to program the robot without using the teach pendant or writing a single line of code. In order to configure the settings for pick and place points, for the palletizing mosaic layout, for grippers and robots, were created, inside this GUI, buttons and text input fields responsible for storing these configurations. The code for the robot is generated automatically after setting all configurations. The processes mentioned above form the ”Palletizing Routine Generation Module”, which, on its turn, will exchange information with the “Robot Trajectory Generation System”, as illustrated in Figure 1. The trajectory generation system aims to program the robot movements between the pick and place points, considering operations within the modeled cell, in a way that results in collision-free trajectories.

Subsequently, 3D simulation acts towards system validation. By simulating the robot’s virtual models and the palletizing systems in which they operate, there may be a significant improvement in the design efficiency of a robotic cell. Also, considering the accuracy and level of detail offered by the VC, it is possible to achieve models having a behavior very similar to the real equipment; this permits studying and testing a solution, with reliability, before its implementation. After simulation validation, the code, written in the VC’s software

proprietary language (Robot Simulation Language - RSL), is translated into the ones used by the different robot's controllers (ABB, KUKA, FANUC, and Yaskawa). For the translation process, the "Translation Module" presented in [15] was used.

III. System Architecture

As presented, the AdaptPack Studio is divided into three modules, each responsible for handling a corresponding subproblem. This paper focuses on the "Palletizing Routine Generation Module". Within this module is included the "Robot Trajectory Generation System". In the sequel, there is an explanation of the developed module in more details.

The VC interface is divided into tabs, each one responsible for a specific task, such as layout construction, components modeling/edition, programming, and others. A new tab, named "ADAPTPACK", has been inserted between the existing ones (see Figure 2) to represent the "Palletizing Routine Generation Module". This new tab is divided into subgroups, that allow configuring the simulation intuitively. The subgroups, as well as their functionalities, are the following:

- Palletize Settings: This subgroup contains a collection of buttons and text fields that define the settings for the product palletizing. The Pick, Pallet, and Interlayer points are selected by clicking on the correspondent 3D component where the products, pallets, and interlayers are stored, respectively. The place point is selected by clicking on the component defined as the drop location. Since a gripper may have multiple Tool Center Points (TCP), it is also present a menu with all the robot tools for this characteristic. Furthermore, there is an option that determines the methodology for the definition of the pick and place approach points, in agreement with the sort of gripper used. The ones that have a side approach (involving toothed grippers), or the ones that have a top approach (for example, suction grippers), are covered in this scope. Finally, it is a good practice for palletizing tasks to determine an approach point just before the grasping/dropping point - this point is automatically generated by entering the distance in a provided text box.
- Mosaic Settings: This subgroup allows the operator to select an input file containing settings for the palletizing tile. Written in the JSON scripting language, this file is generated automatically by external software, given settings such as pallet and product dimensions, the number of layers, usage of interlayers, and others. The imported file consists of structures that determine the order and positions of the objects, in the space of the pallet. It also contains information about the number of layers and dimensions of the products and pallets. The overall file structure has divisions by layers, each layer containing one or more groups, and each group including one or more products. Consequently, the robot can simultaneously handle one or multiple products, depending on the mosaic layout defined by each customer.
- Simulation Settings: Within this area, there is the possibility to perform adjustments to the simulation environment, such as robot's and conveyor's speeds, stop times, product production rate time.
- Path Planning: This area has buttons that permit the activation of the "Robot Trajectory Generation System", based on the A* search algorithm developed in [16]. Besides the discretization in joint space [14], a Cartesian space one was introduced in this work. This new approach allowed decreasing the computation time and the smoothing of the trajectory.

After performing all the configurations, the system is ready to calculate the automatic generation of the palletizing routine. When the "Init Routine" button is activated, the "Palletize.cs" class receives the configurations from the GUI and the selected JSON file, as pictured in Figure 3. The pseudocode presented below shows, step by step, the computation steps that occur inside the class.

```
void Init():

    get simulation settings from GUI
    run simulation for x seconds
    WhenSimulationStops()

void WhenSimulationStops():

    ## Settings
    get info from selected components in GUI
    get picking and placing distance values
    get info about products and pallets
    get components dimensions
    get transformation matrices/pick points

    define tools to be used
    define robot base frames

    ## Robot program
    if(handlePallet enabled):
        define pallet picking and placing routine

    if(handleInterlayers enabled):
        define interlayers picking and placing routine

    define subroutines for controlling signals

    ## Path Planner routine
    if(A* enabled):
        call PathPlanning.cs class
        for(each point in returned trajectory):
            generate path routine

    ## Picking and placing routine
    for(each layer in json file):
        for(each group):
            get tool
            define pick and place points
```

Script 1. Pseudocode for the class Palletize.cs

After initializing, the module simulates the mobile components (boxes and pallets) running on conveyors, until they arrive at the specific pick points on the conveyors. After this procedure, are extracted the variables about the 3D center of those mobile components and calculated the precise pick location. Are also extracted information about height, length, width, and the position/rotation matrices of all components. For generating the placing positions, according to the mosaic, the system interprets the content of the JSON file and calculates the point where to drop products on the provided drop

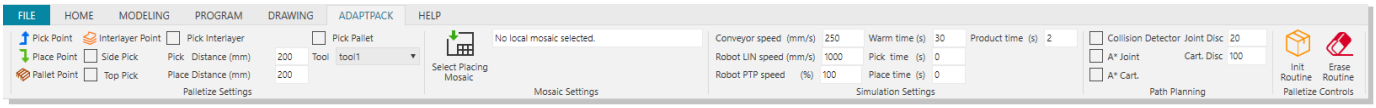


Fig. 2. The GUI subdivided in Palletize, Mosaic, Simulation and Path planning. Here contain all the options required for automatic routine generation.

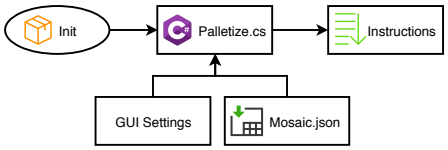


Fig. 3. Palletize class details. The init button initializes the Class, which has external inputs and generates the robotic routine.

location. This interpretation is made recursively until each group has its placing location in the simulation.

The "Robot Trajectory Generation System" was implemented as a feature inside the "Palletizing Routine Generation Module" and, if activated, checks possible collisions in the trajectory, between the pick and place points. This system was build inside a class called "PathPlanning.cs" as illustrated in Figure 4. Two Path Planners are included: (i) using the previous A* discretized in the joint space [14], or (ii) using the A* in the cartesian space. Using the collision detector, provided by the API inside the VC, were produced validations about collision-free paths. Following the A* logic, the simulated cell is decomposed in units. If a unit features a collision between components, it is automatically discarded, and the next one is checked.

Furthermore, within the simulation scope, a custom component was modeled with the purpose to simulate factory Programmable Logic Controllers (PLC). This PLC has connections with robots and other components in the cell. To further detail the simulation, the "Palletizing Routine Generation Module" generates robot operations and also creates I/O signals from all components to this PLC. This custom component should be programmed manually to have the logic to deal with the I/O signals. Different clients may use different types of cell controllers for different demands; thus the programming of those should be made accordingly to different needs.

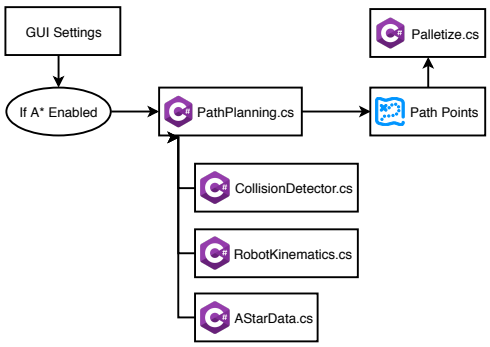


Fig. 4. The "PathPlanning.cs" class details. If activated in the GUI it generates the path to the Palletize class.

IV. Tests and Results

In this section, the performed tests regarding the "Palletizing Routine Generation Module" and the "Robot Trajectory Generation System" are introduced, alongside with the observed outcomes. It was modeled a typical palletizing cell to verify the system feasibility (Figure 5). This cell has a robotic arm equipped with a custom gripper capable of handling single or multiple objects. Extra equipment, such as fences, conveyors, and feeders were included to allow the simulation to act realistically and present the full capacity of the software to simulate real factories operations.

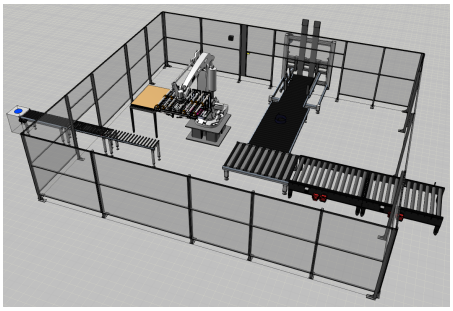


Fig. 5. Representation of the working Cell.

1) Palletizing Routine Generation Module: As explained in Section III, the user must configure the cell, using the GUI, and then press the "Init Routine" button. For this test, was enabled manipulation of products and interlayers. Figure 6 depicts the conveyor from where are picked the products (on the left), the interlayer table (on the top left), and the conveyor where the products are placed (on the right). It is noticed the multiple picking points beside each other on the left conveyor; they were generated according to group handling ability. Sometimes the robot must pick a group containing one product and sometimes a group containing numerous products. In that manner, were calculated the different picking points according to the center of the current product group.

Following the algorithm described in Script 1, tools and base frames are defined, followed by pallet and interlayer routines declarations. The Pick and Place functions were subdivided in subroutines to keep track of the actions the robot is executing, so it always moves in steps. The signals were defined to work along the simulated PLC. This component works as a manager to the complete cell. For example, before the Pick function, the robot interacts with the PLC in order to get permission to continue the operation. The PLC allows the robot to pick products only if the picking conveyor has products ready to be picked. In Figure 7 is shown a snippet of the generated routine. This Figure depicts the Program Editor window, where

the Main routine and subroutines are declared. Inside the Main routine, the tool and the base frames for the robot are defined, followed by the interlayer handling subroutine, and the palletizing script.

2) Robot Trajectory Generation System: Trajectory calculation is an optional resource implemented within the AdaptPack Studio. Without its activation, the robot performs a straight motion from the pick to the place point. Therefore, the "Robot Trajectory Generation System" is suitably used when the cell has tight limits, with the possibility of collisions, or trajectory computation and analysis are necessary. Considering both wide and compact operational spaces, representations of these two scenarios have been modeled to test the Path Planners. There are no collisions in the first scenario (ample space), different from the second (compact space). It was tested and compared both planners considering these two scenarios.

For the first scenario, the trajectory results are presented in Figures 8a and 8b. These images show the correlations between the trajectories generated in an open space without obstacles, and it is clearly represented the distinct paths generated by both Path Planners. The Cartesian discretized trajectory resembles linearity, contrarily to the joint based Path Planner. For this test, was used an interval of 300 mm in the Cartesian discretization as a parameter. For the joint discretization intervals of 20º were tested, for each robot joint.

The second scenario represents a tight space with enclosed barriers. Opposite from the previous, this scenario features collisions when Path Planning is not enabled. As illustrated in Figure 9, the robot gripper collides with the safety fences. To solve this problem, both Path Planners were enabled and tested. The results are illustrated in Figures 10a and 10b. In this case, were considered the obstacles, and the gripper no longer collides with objects.

The Cartesian discretization allows a faster calculation time for small discretization steps, along with fewer computational resources. Table I depicts the average time gathered for 1000 path calculations, using each method and distinct discretization intervals. In the first experiment, the trajectory generation in the Joint space was tested, considering discretization intervals from 15º to 40º. In the second experiment, the path planner considering the Cartesian space was tested, for discretization intervals between 100 mm to 400 mm.

Aside from the previous tests, was conducted a real-world test scenario. For this purpose, a cell consisting of a conveyor, a table and a Fanuc 200ic robot was modeled. Within this test case, the robot task was to pick six metal cans at the top of the conveyor and palletize them on the table (Figure 11). Despite the simplicity of the layout, these results validated the work developed after exporting the generated routine to a real system.

V. Conclusions and Future Work

The use of automatic palletizing solutions has been expanding in the FMCG business. Due to the diversity of production, palletizing processes are changing from low assortment with high volume to high assortment with low volume. Often,

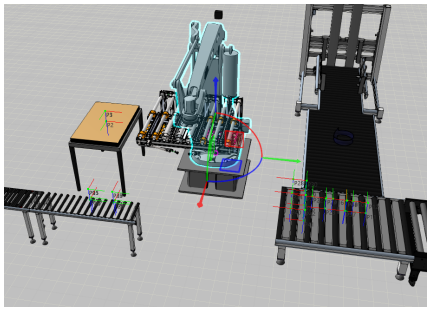


Fig. 6. Generated Pick and Place points without Path Planner and Collision Detector.

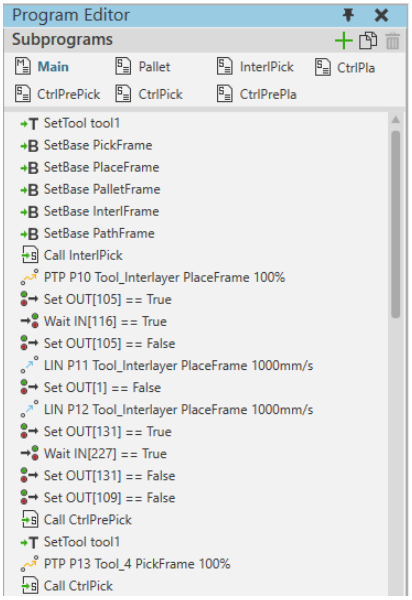
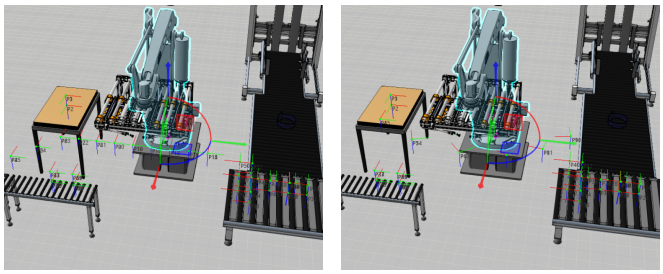


Fig. 7. Generated routine.



(a) Cartesian Trajectory. (b) Joint Trajectory.

Fig. 8. Trajectories generated in an open space.

TABLE I
Discretized samples of Joint and Cartesian space and the time consumed at each step.

Method	Scale	Time	Method	Scale	Time
Joint Space	15º	8.0038 ms	Cart. space	0.10 m	4.0050 ms
	20º	3.0050 ms		0.20 m	2.0057 ms
	30º	1.0046 ms		0.30 m	2.0019 ms
	40º	1.0001 ms		0.40 m	1.0060 ms

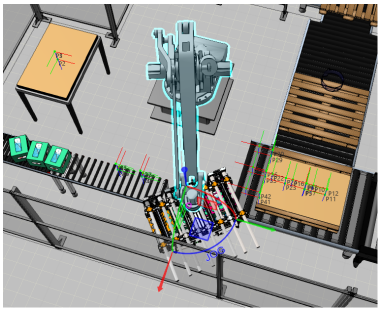
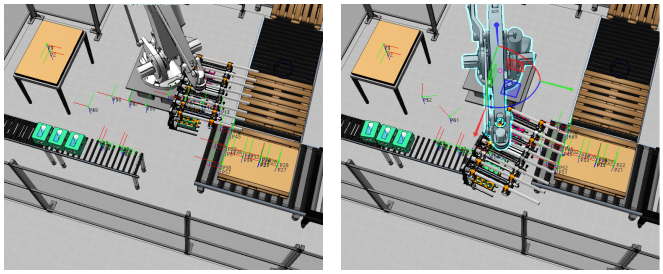


Fig. 9. Representation of a tight space cell. Without Path Planner and Collision Detector the robot collides with the fences.

these changes require modifying machine tasks for different incoming supplies. It implies updating all the programmed routines of robots, as well as changing the palletizing mosaic. Accordingly, this work proposed an answer related to the construction of a qualified framework to enhance automatic palletizing processes and solve preceding issues.

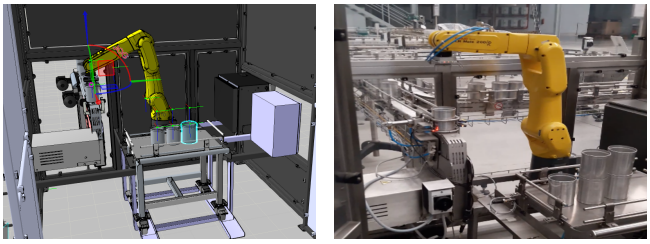
The improvement of a software framework for the automatic offline programming of palletizing applications was presented. This improvement speeds up the task of designing palletizing cells since it allows to model cells from scratch using the functionalities presented in the VC software. Besides, it also enables the integrated and modular development of new component models, reducing the time to design, develop, build, and install new equipment. The automatic generation of robot programs, based on inputs inside a GUI, has been detailed. These inputs consist of clicking at picking and placing positions inside a 3D modeled cell, selecting the mosaic which defines products stacking aspect and writing general configurations in text fields. Also, are included inside this new framework two collision-free trajectory generators, based on the A* informed search algorithm. The first searches a path in the robot's joints space, different from the second, which searches a path in the Cartesian space of a cell. Finally, it was tested the Robot Trajectory Generation System in simulation and in the real world. Based on these preliminary tests, the system works as predicted, being able to generate collision-free automatic palletizing routines.

Notwithstanding, future enhancements must be performed in the A* Trajectory Generator based on Cartesian Discretization. This generator resulted in straight line trajectories, so it is appropriated to smooth this path to something that resembles more the robot's joints natural movements. Furthermore, to conclude the solution validation, there are going to be conducted larger-scale real-world tests.



(a) Cartesian Trajectory. (b) Joint Trajectory.

Fig. 10. Trajectories generated in an enclosed space.



(a) Simulated Cell. (b) Real Cell.

Fig. 11. Simulated and real tests.

REFERENCES

[1] D. I. Miyake, "The shift from belt conveyor line to work-cell based assembly systems to cope with increasing demand variation in japanese industries," *International Journal of Automotive Technology and Management*, vol. 6, no. 4, pp. 419–439, 2006.

[2] M. Hanai, S. Tsuchiya, H. Hibi, T. NakasaiI, and H. Terada, "Development of adaptive production system(aps) to market uncertainty. application to automotive starter assembly line." *Int. Journal of the Japan Society for Precision Engineering*, vol. 33, pp. 1–5, 03 1999.

[3] H. Yu, J. Shan, and X. Zhu, "Off-lineprogramming and remote control for a palletizing robot," in *2011 IEEE Int. Conference on Computer Science and Automation Engineering*, vol. 2, June 2011, pp. 586–589.

[4] Yaskawa. (2019) Motoman robot software. (Accessed: 2019, Mar 26). [Online]. Available: https://www.motoman.com/products/software/

[5] K. Kito, K. Tatsuno, S. Otao, T. Hosotani, and K. Yohino, "A robot controller for a working cell," in *2017 International Symposium on Micro-NanoMechatronics and Human Science*, Dec 2017, pp. 1–7.

[6] H. Cheng and H. Chen, "Autonomous robot teaching using a smart robot in production line," in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec 2013, pp. 1772–1777.

[7] F. M. Moura and M. F. Silva, "Application for automatic programming of palletizing robots," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions*, April 2018, pp. 48–53.

[8] V. Components. (2019) 3d manufacturing simulation and visualization software. (Accessed: 2019, Mar 26). [Online]. Available: https://www.visualcomponents.com/

[9] RoboDK. (2019) Simulator for industrial robots and offline programming. (Accessed: 2019, Mar 26). [Online]. Available: https://robodk.com/

[10] OCTOPUZ. (2019) Robot programming and simulation software. (Accessed: 2019, Mar 26). [Online]. Available: https://octopuz.com/

[11] FlexSim. (2019) Simulation software for manufacturing, material handling, healthcare, etc. (Accessed: 2019, Mar 26). [Online]. Available: https://www.flexsim.com/

[12] Anylogic. (2019) Simulation modeling software tools & solutions for business. (Accessed: 2019, Mar 26). [Online]. Available: https://www.anylogic.com//

[13] SIMIO. (2019) Simulation, production planning and scheduling software. (Accessed: 2019, Mar 26). [Online]. Available: https://www.simio.com/

[14] R. Silva, L. F. Rocha, P. Relvas, P. Costa, and M. F. Silva, "Offline programming of collision free trajectories for palletizing robots," in *Iberian Robotics conference*. Springer, 2017, pp. 680–691.

[15] J. P. Sousa, A. Castro, L. F. Rocha, and M. F. Silva, "Converting robot offline programs to native code using the adaptpack studio translators," in *2019 IEEE International Conference on Autonomous Robot Systems and Competitions*, April 2019.

[16] P. Tavares, J. Lima, and P. Costa, "Double a* path planning for industrial manipulators," in *Robot 2015: Second Iberian Robotics Conference*. Springer, 2016, pp. 119–130.