TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 58, Issue IV, November, 2015

ALGORITHM FOR AUTOMATICALLY GENERATING THE ROBOT PROGRAM FOR A RECONFIGURABLE PALLETISING APPLICATION

Mircea FULEA, Ciprian TANASELEA, Bogdan MOCAN, Mircea MURAR

Abstract: *This paper proposes an algorithm for automatically generating the robot program for a reconfigurable palletizing application, as part of the bigger challenge of developing a reconfigurable palletizing robotic cell, targeted towards SMEs, which should be affordable and as easy to operate as not to require robot or PLC programming skills. An application example, covering all the steps of the algorithm, is also presented.*

Key words: *reconfigurable manufacturing systems, robotized palletizers, automatic robot reprogramming.*

1. INTRODUCTION

The last decade brought a significant increase in personalization needs and a pressure to shorten product life-cycle times; as such, manufacturers are challenged to adopt new product and process technologies, also approaching – among others – the new reconfigurable manufacturing systems paradigm (i.e. reconfigurable machines, cells, processes and / or entire systems) [1] [2]. A reconfigurable manufacturing system, or simply RMS, is a system designed for a quick change in hardware and software structure so that it rapidly adjusts production capacity and functionalities; its goal is to provide ”exactly the capacity and functionality needed, exactly when needed” [3] [4]. In this way, RMSs could support manufacturers in their challenge to customize products and to keep production costs at reasonable levels.

Nowadays most manufacturing systems are programmed for a limited range of activities, foreseen in advance by their developer [5]. Such systems are therefore limited in terms of rapidly adjusting production functionalities (and sometimes even production capacity). They may be called reactive because they react only based on expected (i.e. foreseen) environmental stimulus [5]. To make them suitable to new production functionalities or capacity (not initially foreseen, but requested by an increasingly demanding and unpredictable market [6] [7]), hardware and software adaptations have to be made.

Considering robotised manufacturing applications, such adaptations do require skills in either robot or PLC programming (or many times both) or in using specific configuration management software. Apart from programming, expertise may be needed also in robot and task calibration. Such skills and expertise may not be always available in SMEs (or not worth being available, in terms of financial effort) [8].

Robotised palletising applications, which are of concern for this research, do face the same challenge, although (at least apparently) there is little variation in the (pick-and-place) tasks they perform. However, adapting the packaging process for a new version within a product family or for a new package of the product may not be straightforward. Although not be a problem for large production volumes, where a standard palletising cell may be simply added (or an existing one might be adapted to the new package size), for SMEs it may be just too expensive, especially as production volumes for

each product version might decrease due to extending the product family. Flexible robotic palletising cells (providing a number of "input" lines from which a "shared" robot picks packages and palletises them) might not be an optimum solution as well, because – due to market dynamics – one can never foresee the packaging needs of the customer (for instance the number of product "variants" may exceed the number of input conveyors). Such flexible cells are also more expensive, thus making them less attractive for SMEs.

Reconfigurable palletising robotic cells may be an effective solution to the problem just highlighted. They should be able to automatically adapt to changes in input package size and use different pallet types (while still optimally placing packages on the pallet). From a hardware perspective, developing a reconfigurable robotic cell should not be difficult, as little adaptation is needed for its elements (e.g. controllable conveyor guides, destackers able to handle multiple pallet types or adequate grippers). From a software perspective, things are far more complicated, as pick-and-place logic may totally change, this being reflected in both cell control and robot program.

This paper proposes an algorithm for automatically generating the robot program for a reconfigurable palletising application, as part of the bigger challenge of developing a reconfigurable palletiser, aimed at SMEs, which should be affordable and as easy to operate as not to require robot or PLC programming skills.

The rest of the paper is structured as follows: Section 2 further discusses the problem and related work, Section 3 presents the algorithm (and its development methodology) and the user interface, Section 4 presents an application example, and Section 5 draws the conclusions and discusses future work.

2. THE PROBLEM; RELATED WORK

The approach envisaged by this paper for managing changes within a reconfigurable robotic palletiser, such as changes in package dimensions or needs for different packaging layouts, is illustrated in Figure 1 (the problem domain of the proposed algorithm being marked in Figure 1 by the dotted line).

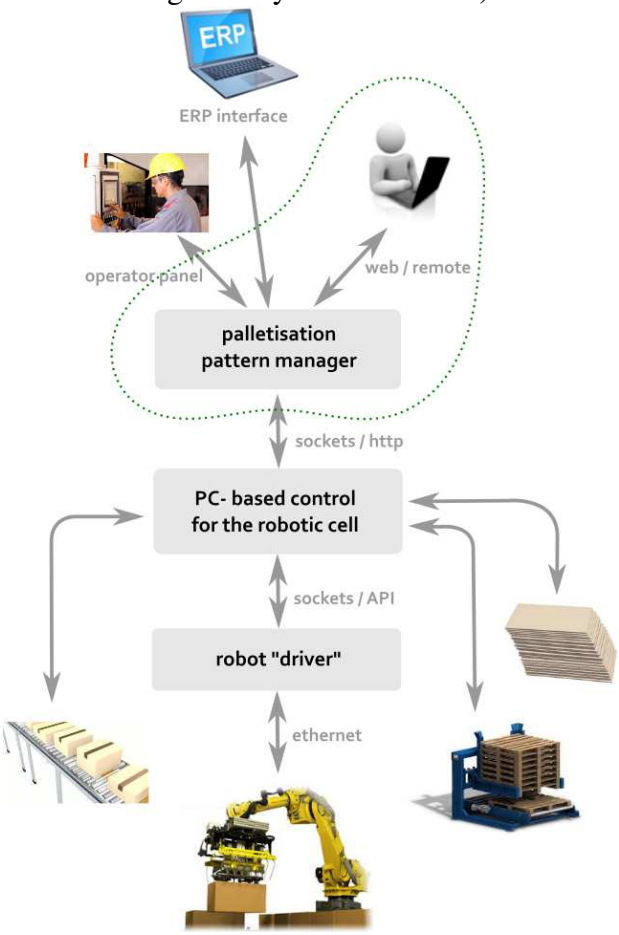


Fig. 1. The control process of the reconfigurable palletiser.

The reconfigurable palletiser has to be interfaced with the connected processes (which supply packages to the reconfigurable palletiser), to be able to detect changes that may occur. Changes could be also be triggered by a user (or operator) or even by an ERP module (which can configure for instance the required product proportions on a mixed pallet). In either case, the configuration data should be passed to a software agent capable of managing the palletising patterns (the way in which boxes / packages are placed on a pallet). The pattern manager should further communicate to the actual cell controller (a computer-based control is suitable for such a cell), which in turn communicates with all the elements of the cell – conveyors and pickpoints, pallet destackers, pallet conveyors, (optionally) cardboard trays and – most important – the robot.

Such communication regards all adaptations that are necessary for instance to arrange boxes

on the pallet in a different pattern or in a different combination (e.g. if they are supplied to the process on two or more input conveyors). Adaptations are needed also if a new input conveyor is added to the process (as the cell is reconfigurable, this should be definitely possible). To handle all these cases, a robot program should handle various points to pick boxes from, various points to place boxes, and possibly various speeds and intermediate points (apart from "randomized" movements if the package flows on input conveyors are not constant). It may be very complicated to build a robot program to foresee all the necessary pick-and-place operations; even using dynamic configuration files may lead to complex robot programs. As such, the authors of this paper explore the possibility of dynamically reprogramming the robot (i.e. automatically generating the program to function until a new change is detected and uploading it into the robot controller).

To identify related work, a survey was conducted, both in scientific databases and on the web, to locate works dealing with the automated reprogramming of industrial robots. The survey also tried to identify papers dealing with reconfigurable manufacturing systems (in general) that use on-the-fly reprogramming or reconfiguration of their elements. The survey results are presented below.

The importance of "knowing" environments to achieve superior flexibility is highlighted in [11]. Being able to be reprogrammed fast by non-robot experts when a new task in the factory arises is also stressed. Intuitive Human–Robot Interaction (HRI) mechanisms are considered to be required for an easy programming of industrial robots. This is in line with the approach hereby; the authors of the current paper consider that, by automatically reprogramming the robot, the user interface can be entirely shifted towards a computer application, which can be easier built to provide a highly usable graphical interface.

The limitations of (contemporary) reactive systems is discussed in [5]; it is noted that a reactive (i.e. pre-programmed) system can be very fragile if something unexpected (like an unforeseen change in the process) occurs and it usually does not have any self-recovery capabilities to prevent or to correct errors aroused from such situations. The importance of context-aware systems is highlighted, as they are able both to act reactively and it can comprehend the present and predict future results or actions. Further on, the paper focuses on the context-aware aspects but not specifically on the reprogramming of the industrial robot.

An interesting mean of automatically generating the robot program is presented in [8], which aims to automatically generate robot motion sequences (programs) from a graphical description of the robot paths over a 3-D CAD model of a given robotic cell, having a general scope of seamlessly integrating product design and robot programming. The automatic generation of a robot program consists of writing robot commands in a text file, line by line, managed by a software interface that extracts data from CAD drawings, interprets it and then generates robot programs.

While further analysing the scientific literature, interesting approaches can be found regarding methodologies capable to generalize robot programs around a task and to be also customized as necessary [9], customization being done in terms of type of robot operation or shape of the workpiece. It is noted that this allows reusing previously similar work, thus incorporating the programmers' experience and process knowledge [10]. Thus, as [8] notes, the time to develop robot programs for related products or tasks can be reduced, enabling also non-specialists to create robot programs.

Further results can be identified in scientific literature to deal with general aspects regarding robot program generation and/or robot program or HRI usability.

2. THE PROPOSED ALGORITHM

This section discusses the algorithm proposed by this paper and its development methodology. It also presents the user interface of the control software that should drive the entire reconfigurable cell, highlighting how information would be transformed from input (user determined, automatic detection of

changes in the process or ERP driven – see Figure 1) into robot software code.

The methodology for developing the algorithm is similar to a Four-Phase QFD planning algorithm [12]. The first step consists of identifying and ranking requirements (using the Analytical Hierarchy Process method – AHP), the second step consists of establishing and planning performance characteristics (using the Quality Function Deployment method – QFD), the third step consists of module and interface planning (using the graphical support of the QFD method), and the last step consists of functionality planning.

The requirements considered for the algorithm are: (RQ1) increased flexibility (24.4%), (RQ2) short execution time (5.7%), (RQ3) low hardware resources requested (4.9%), (RQ4) user interface should be simple (6.4%), (RQ5) easily implementable (8.8%), (RQ6) easily reusable code (14.9%), (RQ7) increased automation (8.8%), (RQ8) scalable (13.2%), (RQ9) well documented (6.4%), permanent feedback on it's state (6.5%). The values within brackets represent the requirement weights (calculated with the AHP method).

The performance characteristics considered for the algorithm (and planned with the QFD method) are listed in Table 1 (M meaning measure unit, O – optimisation trend, T – target value, W – weight or importance).

The performance characteristics were further translated into modules and interfaces and eventually into functionalities (by using the graphical support of the QFD method). The functionality list is as follows: (f1) manage user projects (e.g. palletise a specific product / package), (f2) manage inputs, (f3) manage outputs, (f4) update configuration (package / pallet sizes, flows on input conveyors), (f5) generate pattern list, (f6) display pattern list, (f7) manage layer stack(s), (f8) set specific pattern(s) as active, and (f9) generate robot program for active patterns. Screenshots of the graphical interface are presented in Figure 2 (see the next section).

Table 1

The performance characteristics for the algorithm.

Performance characteristic	M	O		W
number of generated patterns	-	↑	50	10.8%
avg. used pallet surface	%	↑	97	7.7%
pattern generation time	s	↓	<1	3.6%
box length range (diff.)	mm	↑	250	9.1%
box width range (diff.)	mm	↑	250	9.1%
box height range (diff.)	mm	↑	250	8.9%
predefined pallet types	-	↑	4	6.7%
data input avg. time	s	↓	15	5.7%
additional parameters requested from the user	-	↓	0	3.7%
% of hardware resources used	%	↓	3	3.6%
robot types on which the algorithm can be adapted	-	↑	4	4.8%
coordinate calculation errors (for 100 dynamic changes)	-	↓	0	11.3%
no. of bugs (for 100 runs)	-	↓	0	3.8%
„in" interfaces (services offered to higher architectural levels)	-	↑	1	6.9%
„out" interfaces (services offered to lower architectural levels)	-	↑	1	4.2%

The first screenshot shows how a pattern (for arranging one layer of boxes on the pallet) can be managed. The list shows all available patterns (automatically computed, based on the current configuration), and for each pattern the view shows the packages and their coordinates, along with their orientation (one may want to rotate / flip them to achieve a specific label orientation pattern). The placing order is also highlighted, so the exact number of robot movements (and thus the approximate time for placing a layer) is known.

The available patterns can be sorted by the space used on the pallet or by the number of required robot moves, so the entire process can be optimized regarding space usage efficiency or time (or both). A "best" pattern can also be automatically selected by the software.

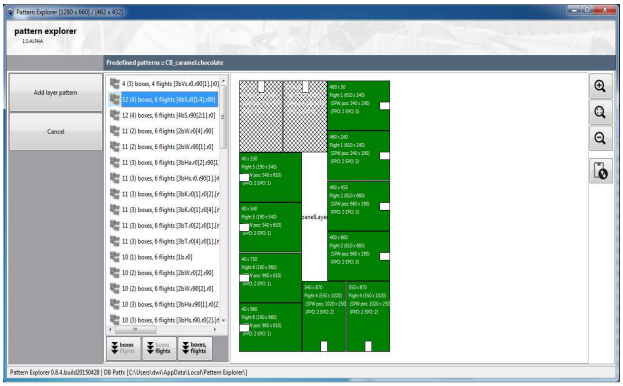
The second screenshot shows how a layer stack can be built (either manual, by the user, or automatically, based on settings).

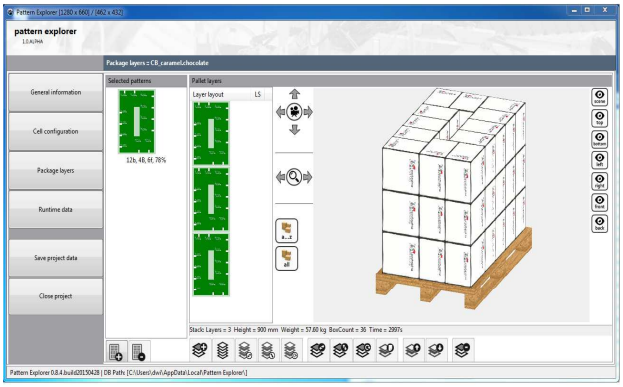
The interface permits layer rotation and mirroring and it shows a 3D view of the package stack.

Based on the automatically computed package coordinates (on a pallet) and on a predefined robot program template, the software can produce the robot program for a specific package stack (either automatically generated or customized by the user).

4. APPLICATION EXAMPLE

This section presents the core use case of the proposed algorithm, in which – based on a package size set and on some standard parameters – an optimum palletisation order is computed and the code for the robot pick-and-place operations is automatically generated.






```
MODULE Module1

CONST robtarget Target_16:=[[-
   155.566039694,1514.646131564,615.761003009],[0.000000016,-
   0.000000044,1,0.000000055],[1,0,1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
CONST robtarget Target_1:=[[1423.038985388,-
   229.440895622,1264.761018386],[0.000000008,-0.000000016,1,-
   0.000000002],[-1,0,-1,0],[9E9,9E9,9E9,9E9,9E9,9E9]];
...
CONST robtarget Target_1_14:=[[825,1189,615],[0.000000008,-
   0.000000016,1,-0.000000002],[0,-1,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

PROC main()
     Path_30;
ENDPROC

PROC Path_30()
   MoveL Target_1,v1000,z100,gripper\WObj:=box;
   MoveL Target_2,v1000,z100,gripper\WObj:=box;
   ...
   MoveL Target_1_4,v1000,z100,gripper\WObj:=box;
   WaitTime\InPos,pi;
   ...
ENDPROC
ENDMODULE
```

Fig. 2. The selected pattern and the automatically generated code.

The size of the package considered for this application example is 250 (width) x 350 (length) x 280 (height) (mm). A stack of three layers was considered. The algorithm generated 23 possible patterns for which the space on the pallet would be more than 80% used and selected the "best" pattern (for which the number of robot moves is as low as possible and the space filled on the pallet is as maximum as possible). The results are presented in Figure 2 (the screenshots).

The coordinates of the boxes were then automatically computed. Using a built-in template for generating the robot program, the code with excerpts shown in Figure 3 was then generated. The robot target points corresponding to box positions on the pallet were simply "translated" from the pattern coordinates, but also intermediate points were automatically calculated to generate an adequate robot trajectory.

To conclude, apart from specifying the package size, no user intervention was required in order to obtain the robot program code.

CONCLUSIONS AND FUTURE WORK

As the results in the application example section show, the proposed algorithm is able to automatically generate the robot program for a reconfigurable palletising application (the only user input, in this case, being the package dimensions). The problem under consideration is just a part of the overall technical challenge of reconfigurability; apart from this, hardware aspects such as plug&play for connecting devices at runtime (e.g. another input conveyor) or control of all elements (e.g. guides on conveyors, pickpoints, pallet destackers) have to be also considered.

One of the advantages of the current approach is that software complexity is being moved from robot software towards "traditional" (PC-based) software, which is much more flexible and easy to develop. As such, user interfaces can be much more productive. The proposed solution can be easily implemented using a common software development environment; although alternative solutions do exist – for instance having a robot program that reads its necessary input data via text files – such a program may become complex and difficult to maintain or extend.

Handling dynamic inputs and outputs, in such an approach, would also be difficult.

Future research will focus on developing a complete control system of a reconfigurable palletizing robotic cell, for which the algorithm proposed in this paper will be part of.

ACKNOWLEDGEMENTS

Support within the project POSDRU/159/1.5/S/137516 PARTING is acknowledged with gratitude.

REFERENCES

[1] Wiendahl, H., ElMaraghy, H. et al, *Changeable Manufacturing - Classification, Design and Operation,* Annals of CIRP 56 (2), pp. 783-809, 2007

[2] Siddiqi, A., de Weck, O., *Reconfigurability in planetary surface vehicles*, Acta Astronautica 64, pp. 589-601, 2009

[3] Fulea, M., Popescu, S., Brad, E., Mocan, B., Murar, M., *A Literature Survey On Reconfigurable Industrial Robotic Work Cells*, Applied Mechanics and Materials 762, pp. 233-242, 2014

[4] ***, What is Reconfigurability, http://www.igi-global.com/dictionary/ reconfigurability/24734, accessed Aug. 2015

[5] Stipancic, T., Jerbic, B., Curkovic, P., *A context-aware approach in realization of socially intelligent industrial robots,* Robotics and Computer-Integrated Manufacturing 37, pp. 79-89, 2015

[6] Zhang, M., Zhao, X,, Qi, Y, *The effects of organizational flatness, coordination, and product modularity on mass customization capability*, Int. J. Production Economics 158, pp. 145-155, 2014

[7] Piller, F., Blazek, P., *Core Capabilities of Sustainable Mass Customization. In: Knowledge-Based Configuration: From Research to Business Cases,* Elsevier Inc, ISBN: 978-012415817-7, pp. 107-120, 2014

[8] Neto, P., Mendes, N., *Direct off-line robot programming via a common CAD package*, Robotics and Autonomous Systems 61 (8), pp. 896-910, 2013

[9] Freund, E., Luedemann-Ravit, B., *A system to automate the generation of program variants for industrial robot applications*, IEEE/RSJ Int. Conf. on Intelligent Robots and System, pp. 1856–1861, 2002

[10] Chen, H., Sheng, W., *Transformative CAD based industrial robot program generation*, Robotics and Computer-Integrated Manufacturing 27 (5), pp. 942–948, 2011

[11] Pedersen, M.R., Nalpantidis, L. et al, *Robot skills for manufacturing: From concept to industrial deployment,* Robotics and Computer-Integrated Manufacturing, in press, 2015

[12] Brad S., Chioreanu A., Fulea M., Mocan B., Brad E., *Reconfigurability Function Deployment in Software Development*, Informatica Economică 15 (2), 2011

ALGORITM PENTRU GENERAREA AUTOMATĂ A PROGRAMULUI PENTRU ROBOT ÎNTR-O APLICAȚIE RECONFIGURABILĂ DE PALETIZARE

Rezumat: Lucrarea propune un algoritm pentru generarea automată a programului robotului în cadrul unei aplicații reconfigurabile de paletizare, ca parte a provocării mai ample de dezvoltare a unei celule robotizate reconfigurabile de paletizare, destinată IMM-urilor, care ar trebui să fie atractivă atât financiar cât și ca ușurință în operare (fără să solicite cunoștințe de programare a robotului sau a PLC-urilor). Lucrarea prezintă și un exemplu aplicativ care acoperă toți pașii algoritmului propus.

Mircea FULEA, PhD, Assoc. Prof, Technical University of Cluj-Napoca, Department of Design Engineering and Robotics, mircea.fulea@staff.utcluj.ro, +40 264 401 766

Ciprian TANASELEA, BSc, Technical University of Cluj-Napoca, Department of Design Engineering and Robotics, ciprian.tanaselea@gmail.com, +40 264 401 766

Bogdan MOCAN, PhD, Assoc. Prof, Technical University of Cluj-Napoca, Department of Design Engineering and Robotics, bogdan.mocan@muri.utcluj.ro, +40 264 401 766

Mircea MURAR, PhD Student, Technical University of Cluj-Napoca, Department of Design Engineering and Robotics, mircea.murar@muri.utcluj.ro, +40 264 401 766