Proceedings in Manufacturing Systems, Volume 13, Issue 3, 2020, 127-133 ISSN 2067-9238

SOFTWARE DEVELOPMENT FOR OPTIMIZING THE PALLETIZING PROCESS OF KLT CRATES

Victor IOSUB[1,*], Adrian Florin NICOLESCU[2], Cristina PUPĂZĂ[3]

[1)] Eng.. PhD Student, Robotic Specialist, Development and Production Department, Robital Industrial Supplier SRL, Bucharest, Romania
[2) 3)] PhD, Prof., Robots and Manufacturing Systems Department., University "Politehnica" of Bucharest, Romania

Abstract: *The paper presents the current development of a new software interface for optimizing the palletizing process of the KLT crates. The algorithm generates a text file with the coordinates of each box that will be placed on the pallet. Furthermore, the file is than loaded in the robot controller for changing the palletizing format. The novelty of the research consists in the software developed using Visual Studio IDE and programmed in C# language. At this stage of the development, the resulting file is loaded in the offline programming and simulating software for Kawasaki Robots, K-ROSET. The research also comprises an overview of the concepts, a test case, and a virtual robotic cell designed in K-ROSET. The virtual model of a palletizing cell was also included.*

Key words: *palletizing software, optimizing palletizing, offline simulation, C# language, algorithm.*

1. INTRODUCTION

Palletization is the operation of arranging in a volumetric order, horizontally – in the form of layers with homogeneous height and vertically – in the form of multiple layers on transport devices called pallets, of different categories of objects, such as: products packed in cardboard boxes with parallelepiped shape, bags with light materials such as granules or powders, sets of multiple pre-wrapped objects – water / oil bottles, etc., handled individually or in groups by industrial robots or automatic palletizing machines [1]. In order to facilitate the unification of storage, transport and handling conditions, the palletizing operation is performed on pallets with dimensions controlled by international standards.

Small load crates (Kleinladungsträger – KLT) are mainly used in the automotive industry for easy storage and transport of the components. The containers were developed with the aim of unifying and standardizing load carriers for automatic handling on conveyor lines. As the automotive industry is on a continuous growth there is a special need for fast implementation times of new palletizing patterns.

2. STATE OF ART

Literature overview on this topic clearly proved that big companies already worked on optimization and generalization of the palletizing procedures, but none has developed a general solution that can work with all brands of industrial robots.

[*] Corresponding author: 313 Splaiul Independentei Avenue, Bucharest, RO-060042
Tel.: +40726499591;
E-mail addresses: *iosubvictor@gmail.com* (V. Iosub), *afnicolescu@yahoo.com* (A. F. Nicolescu), *cristina.pupaza@upb.ro* (C. Pupăză)

Palletizing algorithms deal with the so-called pallet loading problem (PLP). The goal is to determine the optimal placement of a set of parallelepiped boxes on a rectangular surface (pallet). The problem involves placing a maximum number of rectangular boxes in a single rectangular pallet. In an intensive numerical experiment that included 196,557 instances Birgin et al. [2] obtained the optimal solutions in 99.5% of the cases, while the computation time per instance was only 4 seconds. For the MPLP problem, when placing a maximum number of boxes on a pallet, geometrical restrictions must be considered. The boxes have to be orthogonally arranged – the sides of the boxes are parallel to the sides of the pallet – and they do not have to overlap. The boxes can be rotated by 90°. The vertical orientation of the boxes is fixed. Then, after solving the reduced problem (in 2D dimension), the 3D problem can be implemented. Despite its simple structure, based on the work of Fowler et al. [3], the MPLP problem was subsequently classified as an NP-complete problem by Dowsland [4]. Other authors have adopted this classification [5, 6]. When considering the decision version of the pallet loading problem and answer the question whether n boxes can be placed, most of the existing algorithms perform a search through a tree of possible loading patterns. Thus, it brings an exponential complexity and a worst-case scenario in terms of the execution time. On the other hand, there are still doubts about the MPLP problem as an NP – complete problem. These uncertainties are sustained by the fact that even in the cases when pallets must be loaded with a large number of small boxes they have to be solved within a reasonable computational time. Apart from these observations, there are also some theoretical arguments regarding the classification of the problem. For example, Nelissen [7] and Alvarez-Valdes et al. [8] have pointed out that solving any instance of the MPLP needs 4 integer parameters: the length and the width of the pallet,

respectively the length and the width of the box, which is uncommon for NP-type problems. Nelissen [7] even doubted that the MPLP problem is included in the NP problem class. If the problem is transformed into a decision problem, so if it is possible to load n boxes on the pallet, assuming that it is only possible to process a fixed percentage of n boxes at a constant time, then the time required to resolve the input string for a Turing machine is not polynomial. Thus, Nelissen concluded that the complexity of the MPLP problem is unknown. In fact, this is the research level also achieved by Birgin et al. [9], Letchford and Amaral [10], Martins and Dell [11], Morabito and Morales [12], Young-Gun and Maing-Kyu [13].

At present, most of the software packages that optimize the robotic palletizing processes offer, in addition to the creation of the optimal pallet loading scheme also the computation of the optimal dimensions of the secondary packaging, based on the size of the products that have to be palletized, as well as the optimal loading of the container, or the transport truck, respectively.

Thus, these packages have become complex utilities through which the entire logistics chain of a company can be generated, thus dramatically reducing the transport and packaging costs. However, even though the software procedures greatly evolved, still none of them generate the optimal solution for uploading. This is due to the pallet loading problem (PLP) is an NP-tough problem that means an acceptable non-deterministic polynomial problem. All the developed packages use heuristic algorithms. Therefore, there is a great need of investigation in this field to improve the existing algorithms or for developing new algorithms that can solve the PLP problem by obtaining solutions close to the optimal solution in a reasonable amount of time. Furthermore, the software packages that are available on the market also offer the possibility to calculate container fill with pallets. The deep drawback of these packages is that they do not allow a direct interface between the optimized solution and the robotic palletizing system. Figure 1 illustrates one of the software packages on the market.

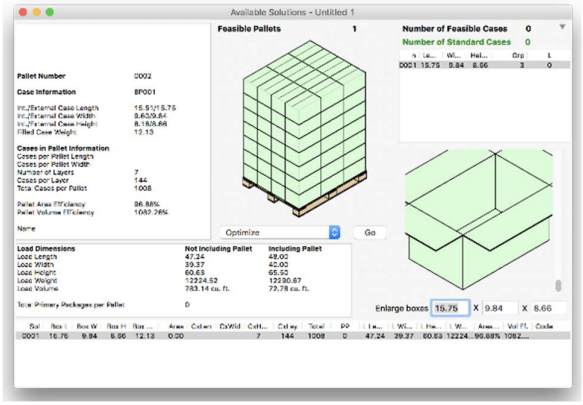


Fig. 1. Example GUI of Quick Pallet Maker 4.8.3 [14].

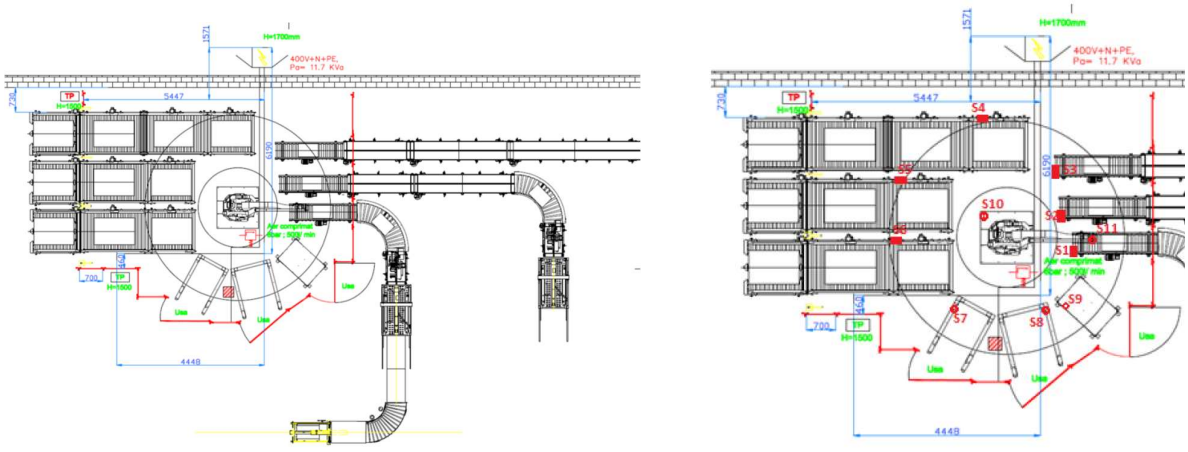
Since actual solutions emerged from big enterprises, they are not affordable for small and medium companies. The current procedure is focused on universal boxes and not devoted only to KLT crates that are so widely used.

The main advantage is that it gives the opportunity to determine the optimal loading of the pallet starting from both the dimensions of the boxes that need to be palletized and the dimensions of the used pallet.

3. PALLETIZING CELL DESCRIPTION

The layout of a robotic palletizing cell was completed in a 2D format in Draft Sight software and the 3D model was generated in SolidWorks 2018 environment. This cell was employed to test the actual development of the original interface. A circular arrangement of the pallet and separator storages around the robot was chosen. The system includes (Fig. 2):

- semi-automatic cardboard box forming machine;
- manual box loading station;
- an automatic carton closing machine;
- automatic printers;
- transport & transfer systems in the form of straight and curved roller and belt conveyors;
- final roller inlet conveyors;
- Kawasaki CP180L robot;
- an Europallet stacker;



a b

Fig. 2. Robotic cell layout: *a* – general layout; *b* – details on the main component location.

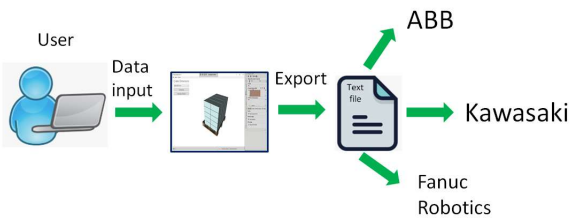


Fig. 3. Schematic view of the concept.

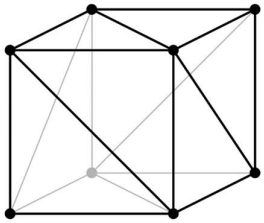


Fig. 4. 3D generation of a cube.

- a block pallet stacker;
- separator for separators;
- roller conveyors for exiting the palletizing cell;
- a perimeter protection barrier.

Figure 2 shows the overall description of the cell, as well as a detail with the components location.

4. CONCEPTUAL APPROACH

The core idea was to develop a compatible solution with the ones already available on the market, but with reliable improvements regarding the link and the interface between the solution and the robotic palletizing system.

The software was developed aiming its versatility and generality, so that it can work with any robotic brand used in the palletizing system. It is a solution oriented to small and medium sized companies that use industrial robots for palletizing and begin with only KLT crates.

Figure 3 gives a schematic representation of the concept. As it can be noticed the application itself is an interface between the operator and various brands of industrial robots. The user has to introduce first the initial data such as: the dimensions of the KLT crates, the size of the pallet and number of layers. The software then processes the data and respects the constraints necessary to have an adequate load on the pallet and presents the optimal solution.

If the result is satisfactory, the user can export a text file containing the solution and can further upload it to the robot controller. Otherwise, a session restart may be proceeded, and the modification of the initial parameters is launched to reach another variant. The export text file can also be loaded in an off-line programming and simulation software to be verified prior to commissioning on the actual production line.

5. THE PROCEDURE

The software procedure is in the test stage. It is written in C# programming language and uses a Microsoft IDE Visual Studio environment.

Because a 3D display area is required for the optimized solution, a Microsoft scheme – Windows Presentation Foundation (WPF) for rendering interfaces has been employed. This graphical user interface (GUI) consists in two main areas. One is the input data area that offers the operator the capability to set up his constraints and enter the required data and the second area is the 3D area in which the rendered solution is presented.

In the robotic cell (Fig. 2) the crates are already oriented in the same position on the long side of the pallet. Even if the crates are of different dimension types, the orientation on the pallet side is the same. At this stage of the software development the center of gravity of the whole package was not included in the model, but this assumption does not affect the present algorithm.

The first step was to determine how to generate multiple 3D parallelepipeds based on the input data runs throughout the algorithm. Generally, in 3D computer graphics the shape of polyhedral objects is defined by a polygon mesh. The polygon mesh is a collection of vertices, edges and faces. To simplify the rendering process, a triangle face was used because it is the simplest geometrical shape.

For a better understanding of the concept, Fig. 4 presents the generation of the cube faces, with only triangles in the 3D environment. The cube is described by 8 points that define vertices and 6 faces, each one defined by two triangles.

The algorithm starts with the definition of a function that returns the cross product of two vectors (Algorithm 1). The input data for this function are the three 3D points, $p0$, $p1$ and $p2$. The function generates two vectors between point $p0$ and $p1$, and between point $p1$ and $p2$, respectively. Then it also returns the cross product of these two vectors resulting a triangle. Algorithm 1 is an extract of the source code of this function.

Algorithm 1. Code of initial function

```
private Vector3D calculateTriangle(Point3D p0,
Point3D p1, Point3D p2)
{
       Vector3D v0 = new Vector3D(p1.X - p0.X, p1.Y
- p0.Y, p1.Z - p0.Z);
       Vector3D v1 = new Vector3D(p2.X - p1.X, p2.Y
- p1.Y, p2.Z - p1.Z);
       return Vector3D.CrossProduct(v0, v1);
}
```

After the triangle position in the 3D space was performed a material was assigned, to be rendered. For this purpose, another function was developed (Algorithm 2).

The function takes as parameters the three 3D points that correspond to the position of the triangle, creates a mesh between those points, assigns a material (a solid blue color) and adds it to the final model.

The last step is to group all the twelve triangles at the corresponding position, so a parallelepiped object is generated. To create the box, the length, the width and height of the box as well as its origin on all three axes have to be considered as parameters. This last function can basically generate a stack of boxes in a parametric

Algorithm 2. Code of second function

```
private Model3DGroup createTriangleModel(Point3D p0,
Point3D p1, Point3D p2)
{
       MeshGeometry3D mesh = new MeshGeometry3D();
       mesh.Positions.Add(p0);
       mesh.Positions.Add(p1);
       mesh.Positions.Add(p2);
       mesh.TriangleIndices.Add(0);
       mesh.TriangleIndices.Add(1);
       mesh.TriangleIndices.Add(2);
    Vector3D Normal = calculateTriangle(p0, p1, p2);
       mesh.Normals.Add(Normal);
       mesh.Normals.Add(Normal);
       mesh.Normals.Add(Normal);
    Material material = new DiffuseMaterial(new
SolidColorBrush(Colors.Blue));
    GeometryModel3D model = new
GeometryModel3D(mesh, material);
       Model3DGroup group = new Model3DGroup();
       group.Children.Add(model);
       return group;
}
```

Algorithm 3. Code of the third function

```
private double generatebox(int l, int h, int L,
double orgx, double orgy, double orgz)
{
Model3DGroup cube = new Model3DGroup();
Point3D p0 = new Point3D(orgx, orgz, orgy);
Point3D p1 = new Point3D(orgx+l, orgz, orgy);
Point3D p2 = new Point3D(orgx+l, orgz+h, orgy);
Point3D p3 = new Point3D(orgx, orgz+h, orgy);
Point3D p4 = new Point3D(orgx, orgz+h, orgy+L);
Point3D p5 = new Point3D(orgx+l, orgz+h, orgy+L);
Point3D p6 = new Point3D(orgx+l, orgz, orgy+L);
Point3D p7 = new Point3D(orgx, orgz, orgy+L);
//
cube.Children.Add(createTriangleModel(p0, p2, p1));
cube.Children.Add(createTriangleModel(p0, p3, p2));
cube.Children.Add(createTriangleModel(p2, p3, p4));
cube.Children.Add(createTriangleModel(p2, p4, p5));
cube.Children.Add(createTriangleModel(p1, p2, p5));
cube.Children.Add(createTriangleModel(p1, p5, p6));
cube.Children.Add(createTriangleModel(p0, p7, p4));
cube.Children.Add(createTriangleModel(p0, p4, p3));
cube.Children.Add(createTriangleModel(p5, p4, p7));
cube.Children.Add(createTriangleModel(p5, p7, p6));
cube.Children.Add(createTriangleModel(p0, p6, p7));
cube.Children.Add(createTriangleModel(p0, p1, p6));
//
ModelVisual3D model = new ModelVisual3D();
model.Content = cube;
this.viewport3D.Children.Add(model);
}
```

mode, in a 3D environment by just calling this function. Therefore, a 3D visualization of a whole stack on the pallet based on the result of the algorithm can be performed.

As it can be noticed from the source code (Algorithm 3) the function creates the eight vertices of a cube in the correct positions, considering the input parameters and then generates the twelve triangles between the points.

At the end, the cube is placed at the desired position in the 3D space.

To better understand how and when these functions are called, a simplified diagram of the algorithm is illustrated in Fig. 4. The first action is an initialization of all elements (variables, functions, etc.). After pressing the "Generate" button, the packaging algorithm is executed. Then, using the solutions given by the algorithm a generate-box function is called. This one employs the create-triangles and calculate-triangle functions for an appropriate operation. In the end the 3D visualization procedure is performed.

To realistically view the 3D objects the model was rendered adding extra light and a camera (Algorithm 4). This is an initializing section. A comment was left at each line to fully understand the procedure. This is the whole 3D motor of the application.

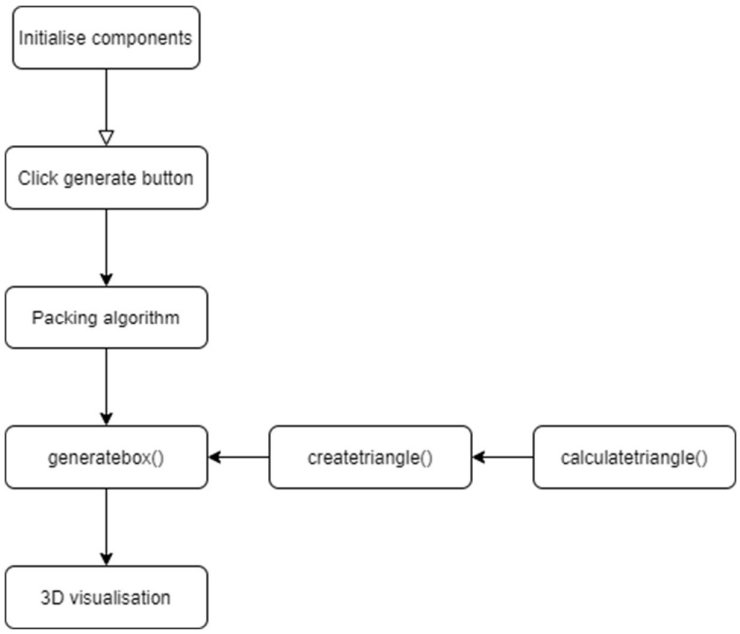


Fig. 4. The algorithm.

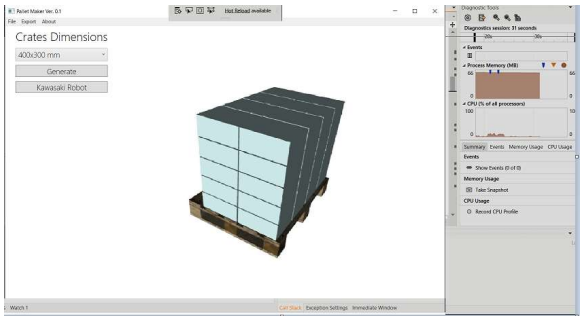


Fig. 5. Application interface.

Algorithm 4. Code of fourth function

```
public MainWindow()
{
       InitializeComponent();
       //Import the pallet 3D model
       ModelImporter importer = new ModelImporter();
       pallet = importer.Load(@"E:\Visual
Studio\PalletMaker\PalletMaker\3D_models\Pallet.obj"
);
       // Defines the camera used to view the 3D
object
       PerspectiveCamera myPCamera = new
PerspectiveCamera();
       // Specify where in the 3D scene the camera
is.
       myPCamera.Position = new Point3D(1, 1, 1);
       // Specify the direction that the camera is
pointing.
            myPCamera.LookDirection = new Vector3D(-
1, -1, -1);
       // Define camera's horizontal field of view
in degrees.
       myPCamera.FieldOfView = 75;
       // Asign the camera to the viewport
       viewport3D.Camera = myPCamera;
}
```

Another stage of the software development was the initialization of the user interface (which is still in progress) (Fig. 5). The interface is split in two main areas: the data input area and the visualization one. For the current state of development, the user needs to choose between the two standard dimensions of KLT crates, the level number of the pallet and enter the crate weight. The weight of the crate is also needed to calculate the total weight of the products and to compare it with the euro pallet loading limit. If the limit is exceeded, the application pops-up an error message telling the user that the number of entered layers is too big for the total weight and this number needs to be changed.

The final step of the development was to establish the appropriate way to export the obtained result. A simple text file was chosen because it is the most convenient format. As such, the application creates a text file in which it writes the marker of the box and the coordinates of the origin for each box, in millimeters, on *X*, *Y* and *Z* axes. An extract of a file exported from the application is:

```
0 0 0 0
1 410 0 0
2 0 610 0
3 410 610 0
...
10 0 610 420
11 410 610 420
```

The data included in the export file has to be previously interpreted, in order to be used in all brands of robot's programming software.

5. SOFTWARE TESTING

At this point of the research a test for the whole concept has been performed. The test resides in the transfer of the application output file to the robot software and the execution of the code.

This has been performed employing the offline programming and simulation software K-ROSET from Kawasaki. For this purpose, a small palletizing procedure was written and executed.

A virtual robotic cell with pick positions and drop positions for the pallets and the crates was designed. The cell includes three picking lines of crates from the input conveyors and three dropping areas, corresponding to the pallet's locations on the output conveyors. The objects that are in the cell were created in SolidWorks and then imported in the simulation software using a STL file, which is a general format that can be interpreted by K-ROSET. The cell was designed to work for simultaneously completion of three different pallet stacks. The IR teach-in is done by the operator one time for first layer from each pallet to obtain the individual palletizing scheme.

The crates are coming into the robotic palletizing cell on three different conveyors from the packaging area. All the conveyors are conveniently aligned next to the robot, which is placed on a pedestal for a favorable positioning scheme of the robot working space versus the served palletizing stations. The robot can handle two types of pallets (Euro pallet and 1200 mm × 1000 mm BLOK pallet) supplied from two dispensing units and place them on the output roller conveyors, in three palletizing station with different specific locations. After the pallets are loaded, the palletizing process is started in respect with the data obtained from the developed software by picking crates from each line and placing them on the corresponding pallet. When the full set of palletizing layers was completed for each pallet, the system evacuates individually the pallets.

Figure 6 illustrates a snapshot of the K-ROSET software with the 3D model of the designed robotic cell. The software has the capability to simulate the whole functionality of the robot, checking the accuracy of the TCP positions obtained from the software, as well as to ensure the virtual commissioning of the entire system.

Another robot software procedure was written to interpret and to employ the data exported from the main PC trough the K-ROSET software. The snapshot bellow is an extract from the robot code that is working based on the file exported from the developed application.

It can be noticed that if the crate number is 0, the position for dropping it is the first position that is manually saved in the robot memory by the programmer.

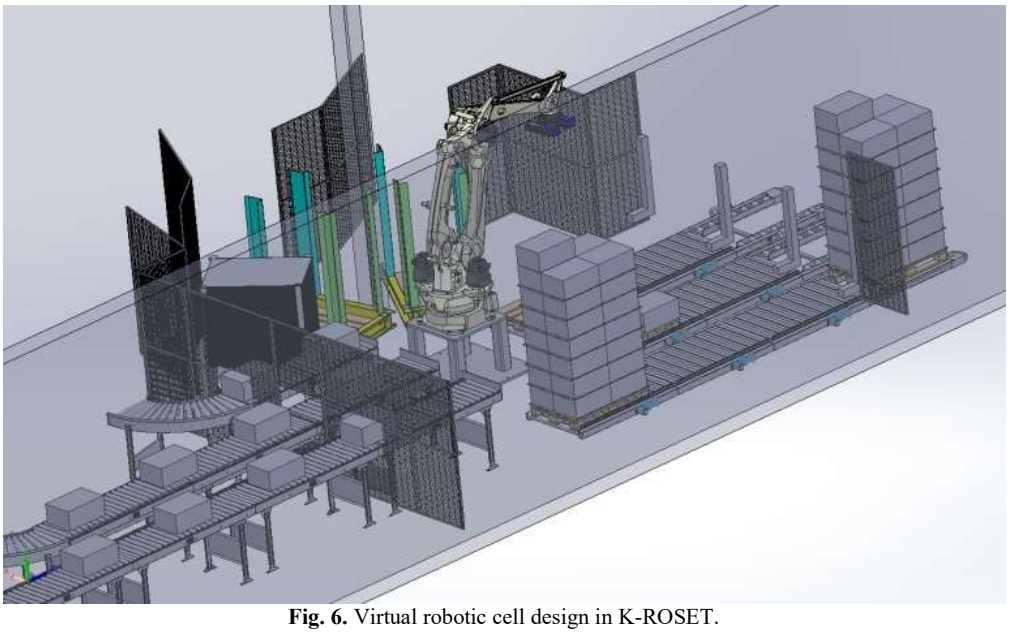


Fig. 6. Virtual robotic cell design in K-ROSET.

Algorithm 5. Extract from robot code

```
LMOVE #preluareL3
TWAIT 0.5
SIGNAL -10,9
TWAIT 0.5
SIGNAL 2008
LDEPART 1500
;
 IF levell1==0 THEN
  IF cutiel1==0 THEN
   POINT depunere = #cutieL1
  END
  IF cutiel1==1 THEN
   POINT place = #cutieL1
   POINT depunere = SHIFT(place BY 410,0,0)
  END
  IF cutiel1==2 THEN
   POINT place = #cutieL1
   POINT depunere = SHIFT(place BY 0,310,0)
  END
  IF cutiel1==3 THEN
   POINT place = #cutieL1
   POINT depunere = SHIFT(place BY 410,310,0)
  END
 END
```

As the execution continues, the data collected in the file start to be used. If the crate number is one, it will be placed in the original first position, but this time shifted on the *X* axis with 410 mm, suggested by the imported file). For the crate number two the placing position is the original one, shifted with 310 mm on *Y* axis. The placing position of the crate number three is the original one, shifted with 410 mm on *X* axis and 310 mm on *Y* axis. A pattern of the execution in progress can be drew-up as shown in Algorithm 5.

When reaching the next layer of the stack, the robot needs to shift the original position on the *Z* axis as well, so it can build the whole stack on the pallet. Bellow it is only an extract from the code for one of the crates of the second layer. Because the height of the crate is 210 mm, the first box of the second layer will increase the position location from the original one to a shifted one, but only on the Z axis, with 210 mm. Algorithm 6 illustrates an extract from the robot programming code for the second layer generation.

Algorithm 6. Extract from robot code for the second layer

```
IF levell1==1 THEN
 CASE cutiel1 OF
 VALUE 0:
  POINT place = #cutieL1
  POINT depunere = SHIFT(place BY 0,0,210)
 any:
 END
END
```

When simulating the functionality of the robot the obtained results proved a correct generation of the of the crates location coordinates. The calculation of the robot TCP corresponding position was accurately reached.

6. CONCLUSIONS

The software procedure presented in this paper is basically developed for box's storage, not only for the KLT crates that are widely used in the automotive industry. The main advantage of the software is that it automatically defines the optimal loading of the pallet starting from the dimensions of the boxes that need to be palletized and the dimensions of the used pallets. The actual developed version of the software operates with the same dimensions and orientation of the boxes for all stack layers and it perfectly fits the cumulated box's dimensions and respectively the pallet dimensions.

Work is in progress to generate the pallet stack for any palletizing scheme of the boxes on each layer. For this purpose a recursive algorithm will be used. Another further development is intended to be done for mixed palletizing which is the most complex problem regarding palletizing. The solution of the problem for this purpose is usually obtained from a heuristic algorithm. Future testing will involve optimized data implementation to a real robotic cell, similar with the proposed virtual model.

REFERENCES

[1] A.F. Nicolescu, *Industrial robots implementing into production systems (in Romanian)*, Politehnica Press, ISBN 978-606-515-915-0, Bucharest, 2020

[2] I.E Birgin, R.D. Lobato, R. Morabito, *Generating unconstrained two-dimensional non-guillotine cutting patterns by a Recursive Partitioning Algorithm*, Journal of the Operational Research Society, Vol. 63, No. 2, 2012, pp.83-200, DOI: 10.1057/jors.2011.6

[3] R.J. Fowler, M.S. Paterson, S.L. Tanimoto, *Optimal packing and covering in the plane are NP-complete*. Information Processing Letters 12, 1981, pp. 133-137.

[4] K.A. Dowsland, W.B. Dowsland, *Packing problems European Journal of Operational Research*, Vol. 56, Issue 10, January 1992, pp. 2-14.

[5] Subir Bhattacharya, R. Roy, Sumita. Bhattacharya, *An exact depth-first algorithm for the pallet loading problem,* European Journal of Operational Research, Vol. 110, No. 3, 1998, pp. 610-625, DOI: 10.1016/S0377-2217(97)00272-5

[6] V. Pureza, R. Morabito, *Some experiments with a simple tabu search algorithm for the manufacturer's pallet loading problem*, Computers and Operations Research, Vol. 33, No. 3, 2006, pp. 804-819, DOI: 10.1016/j.cor.2004.08.009.

[7] J. Nelissen, *How to use structural constraints to compute an upper bound for the pallet loading problem*. European Journal of Operational Research 84, 1995, pp. 662-680.

[8] R. Alvarez-Valdes, F. Parreño, J.M. Tamarit, *Reactive GRASP for the strip-packing problem*, Computers & Operations Research, No. 35, 2008, pp. 1065-1083.

[9] E.G. Birgin, R. Morabito, F.H. Nishihara, *A note on an L-approach for solving the manufacturer's pallet loading problem*, Journal of the Operational Research Society, No. 56, 2005, pp. 1448-1451.

[10] A. Letchford, A. Amaral, *Analysis of upper bounds for the Pallet Loading Problem*, European Journal of Operational Research, No. 132, 2000, pp. 582-593.

[11] G.H.A Martins, R.F. Dell, *Solving the pallet loading problem*, European Journal of Operational Research, Volume 184, Issue 2, 16 January 2008, pp. 429-440.

[12] R. Morabito, S. Morales, *A simple and effective recursive procedure for the manufacturer's pallet loading problem*, Journal of Operational Research Society, No. 49, 1998, pp. 819-828.

[13] G. Young-Gun, K. Maing-Kyu, *A fast algorithm for two-dimensional pallet loading problems of large size,* European Journal of Operational Research, No. 134, 2001, pp. 193-202.

[14] Quick Pallet Maker, available at: https://quick-pallet-maker.soft112.com/.