International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:6, No:5, 2012

Creation of a New Software used for Palletizing Process

Dušan Kravec, Ondrej Staš, Marián Tolnay, Michal Bachratý

***Abstract*—**This article gives a short preview of the new software created especially for palletizing process in automated production systems. Each chapter of this article is about problem solving in development of modules in Java programming language. First part describes structure of the software, its modules and data flow between them. Second part describes all deployment methods, which are implemented in the software. Next chapter is about two-dimensional editor created for manipulation with objects in each layer of the load and gives calculations for collision control. Module of virtual reality used for three-dimensional preview and creation of the load is described in the fifth chapter. The last part of this article describes communication and data flow between control system of the robot, vision system and software.

***Keywords*—**Palletizing, deployment methods, palletizing software, virtual reality in palletizing.

I. Introduction

PALLETIZATION process has been largely affected by automation. Robotic workstations can deploy objects of different sizes and shapes in high-speed level.

Progressive solutions for palletizing systems are closely linked to development of the vision system used for components identification. Workplace, which is used especially for research of the palletizing process, was established at the Faculty of Mechanical Engineering of Slovak University of Technology in Bratislava. This workplace consists of the robot – YAMAHA YK400X (SCARA construction), belt conveyor, pallets, swap space, control unit RCX 142, computer and vision system [7]. It is used for vision system research, testing different deployment methods of palletizing, measurement of accuracy of robot positioning and mostly for new software development used for palletizing process – *Pallet SjF STU*. The software development started in 2009. We decided to use Java programming language because Java applications are executable on every machine without special hardware requirements.

The only requirement is to install Java JRE (Java Runtime Environment), which contains Java virtual machine and basic Java libraries (Java SE). Java JRE can be downloaded for free from Sun Microsystems, Inc. website (www.sun.com). Basically, the first purpose of the software was robot control.

D. Kravec is with the Institute of Production Systems, Environmental Technology and Quality Management, Slovak University of Technology, Bratislava, Slovak republic (e-mail: dusan.kravec@stuba.sk).

O. Staš is with the Institute of Production Systems, Environmental Technology and Quality Management, Slovak University of Technology, Bratislava, Slovak republic (e-mail: ondrej.stas@stuba.sk).

M. Tolnay is with the Institute of Production Systems, Environmental Technology and Quality Management, Slovak University of Technology, Bratislava, Slovak republic (e-mail: marian.tolnay@stuba.sk).

M. Bachratý is with the Institute of Production Systems, Environmental Technology and Quality Management, Slovak University of Technology, Bratislava, Slovak republic (e-mail: michal.bachraty@stuba.sk).

The next aims were to create flexible two-dimensional editor used for creating layout of objects in each layer and to allow three-dimensional preview of the load.

II. Software Structure

Software structure deals with a data flowing and a parent – child class scheme. The pallet (project) is the main element here. It is divided into sub-tasks: planning each layout of the pallet layers, three-dimensional preview, program generator, data – transmission module (communication with control unit of SCARA robot) and vision system module.

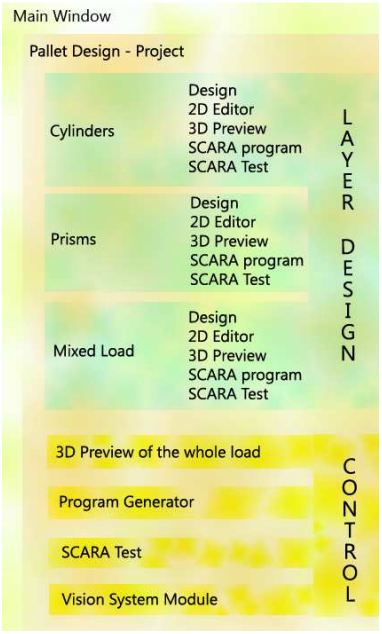

Fig. 1 Software structure

In each layer another deployment method can be used. The automatic layout designer was build for this purpose. It can analyze chosen object in terms of shape and its dimensions. This analysis is able to choose the optimal deployment method according to the given dimensions of the pallet. Several deployment methods for the selected shape of the object are available in manual mode of the layout designer. Two – dimensional editor was compiled for manual moving of the objects. After layout editing, there is three – dimensional preview of the layer available. Program generator is used to create commands for the control unit. Finally, the robot control module sends the generated commands to the robot and belt conveyor.

There is a three-dimensional preview of the whole load available with complete program for control unit after finishing the layout. This information is sent forward to the control module.

This module is in direct connection with control unit of the robot, belt conveyor and vision system that sends data about identifying objects and theirs positions on belt conveyor.

Pallet SjF STU uses *.ppf* (pallet positions file) files for storing and loading data. Robot options are available in the software settings. The connection can be setup and also calibrating of help points is available here for accurate positioning of the robots and effector. Pallet SjF STU is created in two language versions – Slovak and English.

III. Automatic Layout Designer

Several deployment methods for cylindrical and prism shaped objects are implemented in this module. Each deployment method contains algorithms and equations for calculation of object coordinates. Those calculations provide class named "Object Calculations". Input data to this class are the shape of the object, the dimensions of the object and the dimensions of the pallet. Output data from the class are coordinates and orientation of each object on pallet and number of objects.

Only two deployment methods are well known for cylindrical objects [7]:

1) Raster method
2) Cross method

Deployment methods for prism shaped objects are divided according to number of blocks. Each block contains prisms of equal dimensions and orientation (e.g. first block contains prisms with 90° orientation and second block contains prisms with 0° orientation). Deployment methods for prism shaped objects are [6]:

1) Single block method
2) Two block method
3) Three block method
4) Four block methods
    a. Steudls algorithm
    b. Smiths and Decanis algorithm
5) Five block method

Next part of the article contains a short preview of each deployment method with calculations and equations of object coordinates.

A. Single block, 2 block and 3 block method

A method using a single block is the simplest method for resolving the pattern of prisms on the palette. There are only two possible options to place the prisms: vertical or horizontal. Block dimensions are identical to the dimensions of pallets [3].

Calculation of coordinates for horizontally oriented prisms is (1):

$$P_{i,j} = \left[\frac{l}{2} + l(i-1); \frac{w}{2} + w(j-1);\ H \right] \quad (1)$$

With condition: $l(i-1) + l < L$ and $w(j-1) + w < W$

Calculation of coordinates for horizontally oriented prisms is (2):

$$P_{i,j} = \left[\frac{w}{2} + w(i-1); \frac{l}{2} + l(j-1);\ V \right] \quad (2)$$

With condition: $w(i-1) + w < L$ and $l(j-1) + 1 < W$

Where:

L – the length of the pallet *l – the length of the prism*

W – the width of the pallet *w – the width of the prism*

With this conditions: $L \geq W$ *and* $l \geq w$

Method for using 2 or 3 blocks is basically an extension of single block method. In the method of 2 blocks we want to find the best combination of vertically and horizontally oriented prisms along the length of the pallet. Optimum combination contains smallest unused space between prisms. We can also use the third block to fill the empty area above the second columns (columns of vertically oriented prisms) by using inverted prisms [3].

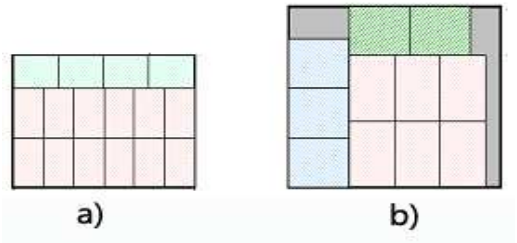

Fig. 2 a) Two-block b) Three-block deployment method [3]

Analogically we can use this method for the width of the pallet and find the best solution for placing the prisms. The advantages of these methods are the simplicity and the clarity. The disadvantage is that it does not provide an optimal solution for every problem [3].

B. Steudls 4 block method

This method divides storage area into four blocks located in the corners of the pallet. These blocks are rotated depending on the direction of stored objects (horizontal or vertical). This method was described by Harold J. Steudl in 1979 therefore it is also known as Steudl algorithm. It is a recursive method and uses dynamic programming. Dynamic programming is used to optimize process. It divides a big problem into a little sub problems. These sub problems are solved and the results for future potential use are stored [7].

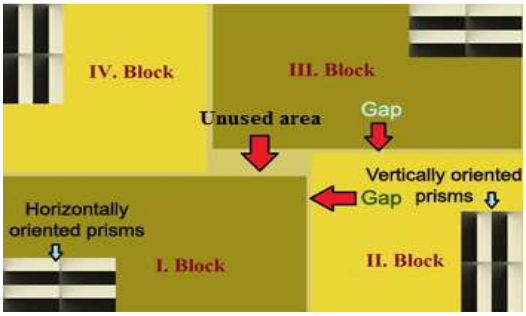

Fig. 3 Steudls algorithm [7]

The first step of this method is finding an efficient combination of horizontally and vertically oriented prisms on a circuit of the pallet. In other words, we are trying to find the smallest gap between combination of horizontal and vertical oriented prisms stored in length (width) of the pallet. We can do this by using this objective function (3):

$$F_n(S_n) = Max\left[X_n * l + Y_n * w + F_{n-1}(S_{n-1})\right]$$
$$X_n * l + Y_n * w \le D_n \qquad (3)$$
$$n = 1, 2, 3, 4$$

Where:

$F_n(S_n)$ – *max sum of horizontally and vertically oriented prisms on the n side with state variable Sn at the beginning of the side.*

X_n – *number of objects of length L placed along the side n*

Y_n – *number of objects of width W placed along the side n*

D_n – *length of the pallet*

S_n – *is a state variable that defines the initial conditions for the side n.*

This state variable can take these three values [7]:

- Prisms are only horizontally oriented along the side *n* of the pallet
- Prisms are only vertically oriented along the side *n* of the pallet
- Prisms are vertically and horizontally oriented along the side *n* of the pallet

Objective function calculates each combination of three values from Sn. We can use this calculation for the three possible values of Sn and four blocks of prisms: $3^4 = 81$. We have to choose the best result from 81 results. Objective function Fn (Sn) can be described as the function that maximizes the utilized length of each side. However, we have to include the pattern of prisms on the previous side. It can be also defined as minimizing the unused circuit of the pallet [7].

In the second step we have to fill the unused area. This step is linked by 2 problems. The first one is filling of empty area, which can hold one or more prisms. The second problem is overlap of individual blocks, which can be identified for example in this case (4):

$$D_1 - X_1 l < X_3 l \quad and \quad D_4 - X_4 l < X_2 l \qquad (4)$$

Steudls method is very good for creating efficient pattern of prisms on pallet but we have to be careful not to create an overlapped area.

C.Smith and Decanis 4 block method

Fig. 4 shows the layout of prisms on a pallet according to Smith and Decanis. It compares all possible combinations of the shown layout.

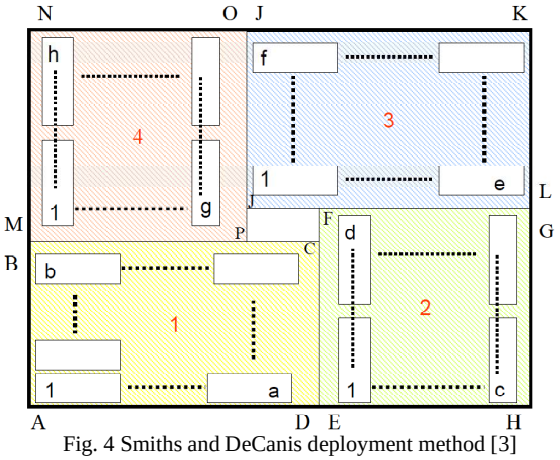

Fig. 4 Smiths and DeCanis deployment method [3]

This method is similar to Steudl method but the number of objects in the blocks 1-3 and 2-4 is not equal. The principle of determining the number of objects across the width and length of each block is different. The first step is to define the first block. The second block needs to be higher than the first block. The third block needs to be wider than the second block and the fourth block is created in the remaining empty space. All possible dimensions of the first block and also dimensions of the other blocks are calculated. Then we can choose the best solution for our dimensions of prisms and pallet. This solution contains the highest number of prisms, which we can store on one pallet [7].

Objective function for this method is (5):

$$MaxZ = a * b + c * d + e * f + g * h \qquad (5)$$

Optimization is finished after generating all possible combinations. This method does not allow overlapping of blocks as it can occur in Steudl method but we can find more unused space between blocks. This problem can be solved either by adding another block or several blocks into this empty space [7].

D.Cylinders

In terms of methodology, the deployment cylinders are the easiest shape for planning the layout on the pallet. In Fig. 5 we can see two simple layouts of cylinders, which are not so difficult. In the first case, there are loses among cylinders. In the next case, we are trying to eliminate these losses but not always successfully. This case is effective only if the number of cylinders in the first row equals to the number of cylinders in the second row. We cannot say which of these layouts is more efficient because sometimes there is the same number of cylinders in the both cases. More advanced software can evaluate both layouts and give the best solution to the user [7].

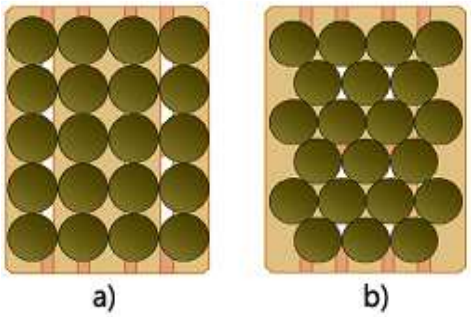

Fig. 5 a) Raster deployment method b) cross deployment method [7]

Calculation of coordinates for „raster" alignment of cylinders (6):

$$P_{i,j} = \left[\frac{d}{2} + d(i-1); \frac{d}{2} + d(j-1) \right] \quad (6)$$

Calculation of coordinates for „cross" alignment of cylinders (7):

$$P_{i,j} = \begin{bmatrix} d - \frac{d}{2}\left(2 * \left[\frac{j}{2}\right] - j\right) + d(i-1); \\ \frac{d}{2} + d * \frac{\sqrt{3}}{2}(j-1) \end{bmatrix} \quad (7)$$

After resolving the question of layout, it is necessary to answer the question of stability. The pads are inserted between the layers of cylinders that enhance the stability of the whole system. It is inappropriate to combine both types of layout between layers. We are using different accessories to ensure stability such as walls, fences, packing washers, belts, etc. [7].

IV. Two-dimensional Layout Manager

Creating object layout by implemented algorithms from software is very limiting. Users mostly want to manually change generated positions of the objects. Two–dimensional editor gives the opportunity to do those changes. It is a separate window with workplace, table and control buttons. Generated coordinates of objects are the input data to this window. New coordinates and orientation are generated for each object after editing and this information is the output from window. Coordinates for each object are written in the table. So number of rows in this table equals to number of objects. Columns are coordinates (x, y and z), orientation (0°, 90°) and index of the object. Index is very important because it identifies each object. New coordinates and orientations are generated after editing positions. Table is updated after clicking on GENERATE POSITIONS button. We have to solve problem with ordering the objects. In Java library there is very good method called *compareTo()*. This method can order objects according to theirs coordinates x and y. Firstly objects are ordered by x coordinate and secondly by y coordinate. The workplace is important part of this window. There is two-dimensional preview of objects stored in layer and outline of pallet dimensions. We can manipulate with objects by using the mouse "drag & drop" method and control keys of keyboard. Objects which require change of orientation (e.g. prism-shaped objects) are rotated after clicking on them by mouse. Each object has its own *JPanel*. All JPanels are managed by *JLayeredPane()* – it is a good control system that allows easy manipulation with objects. Controlling objects by keyboard has more accuracy. SPACE button is used for changing the object selection therefore index of object is very important here. ARROWS are used for manipulation with objects. In some cases it is necessary to input the new object of another shape (in mixed loads) to a workspace. That's possible by clicking on ADD OBJECT button and selecting from menu wanted shape.

After editing the objects positions it is very important to generate new coordinates. There was a small problem because they are read from the screens pixels. Pixel coordination is an integer number. According to accuracy we need decimal numbers of coordinates. Great solution of this problem is to use Affine Transform class with *setScaleTo()* method. It transforms scaling of x and y coordinates. We have to set it on a half (0.5) so it looks like this: *setScaleTo(0.5, 0.5)*.

Very important function of the editor is collision control. It can detect collision with another object in a workplace. Second part of the control is searching for objects, which protrude from pallet. Whole control consists of two ways: *continuous* and *final*.

The control fiber starts automatically after the manipulation with object in the continuous way. If the object is overlapping another object (or is out of pallet) then its color changes from green to red. After changing the selection to another object all objects are painted back to green and the control fiber is calculating collisions according to a new object.

Final collision control starts with clicking on the POSITION CONTROL button. This control calculates positions between all objects on the workplace and checks layout according to stability. Bad situated objects are repainted from green to red. It is necessary to provide new positions for red painted objects. Algorithm for how collision control works for cylindrical-shaped objects, prism-shaped objects and collision control at mixed load on pallet is described in the next part of this article. Fig. 6 shows variables from objects dimensions used later in the equations and conditions.

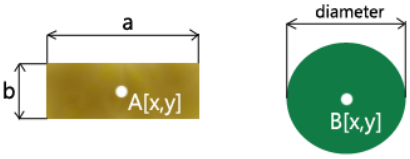

Fig. 6 a) Prism dimensions b) Cylinders dimensions

A. Collision control of cylindrical – shaped objects

Cylinder-shaped objects are usually stored on the pallet only on the flat side of the shape because of the stability. Our SCARA robot has vacuum end-effector so it is very difficult

to catch object at the rotary side. This is why we are planning loads only with possibility of storing cylinders on their flat side. It means that it doesn´t matter what orientation cylinder has. Cylinders are shown as circles with the equal diameter in two-dimensional top-view (Fig. 7) so we can use analytical geometry for calculating distances between them.

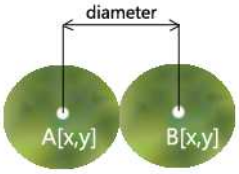

Fig. 7 Collision control of two cylinders

This is an equation for distance calculation between midpoints of two cylinders:

$$distance = (Ax - Bx)^2 + (Ay - By)^2 \quad (8)$$

Ax – coordination x of the middle of cylinder A
Ay – coordination y of the middle of cylinder A
Bx – coordination x of the middle of cylinder B
By – coordination y of the middle of cylinder B

Variable *distance* is very important in collision control between cylinders. If distance is lower than diameter of cylinders then collision occurs. So the condition for safe storage without collision with another cylinder is:

$$distance \geq diameter \quad (9)$$

This algorithm was implemented to the class named "Cylinder", which takes attributes from *JPanel* class. Constructor of "Cylinder" class (object) consists of another attributes – diameter, height, index, color and coordinates(x, y, z). Each object of "Cylinder" class has these attributes. If the collision occurs, all collided objects are red painted.

B. Collision control of prism – shaped objects

Class named "Prism" was created especially for prism-shaped objects and it takes attributes from *JPanel* class from Java JRE library. Constructor of this class includes length, width, height, index, color, coordinates (x, y and z) and orientation attributes. It is possible to rotate objects only around one axis (generally it is z axis) because of SCARA construction of the robot.

Collision control algorithms are complicated because of added orientation attribute. All possible cases of prism-shaped objects orientations are shown in the next figures.

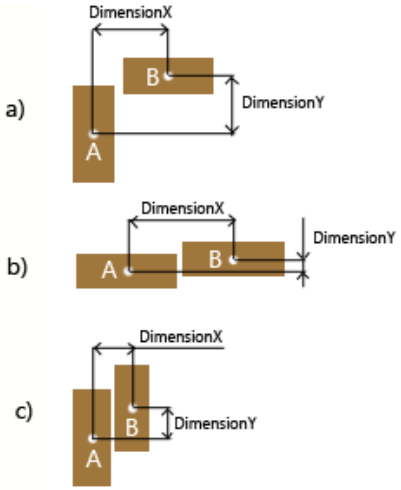

Fig. 8 Collision control between: a) 90° and 0°orientation of the prisms b) 0° and 0° orientation of the prisms c) 90° and 90° orientation of the prisms

Variables *DimensionX* and *DimensionY* are calculated in the first step of the control. DimensionX is absolute value of difference between *x* coordinate of prism *A* midpoint and *x* coordinate of prism *B* midpoint. DimensionY is absolute value of difference between *y* coordinate of prism *A* midpoint and *y* coordinate of prism *B* midpoint.

$$\begin{aligned} DimensionX &= |Ax - Bx| \\ DimensionY &= |Ay - By| \end{aligned} \quad (10)$$

According to these conditions of not being in collision are those two variables evaluated.

In case a)

$$\begin{aligned} &If\ DimensionY \leq a/2 + b/2\ \ and \\ &DimensionX \geq a/2 + b/2 \end{aligned} \quad (11)$$

In case b)

$$If\ DimensionY \leq b\ \ and\ \ DimensionX \geq a \quad (12)$$

In case c)

$$If\ DimensionY \leq a\ \ and\ \ DimensionX \geq b \quad (13)$$

If those conditions are satisfied, then collision doesn´t exist. In the other case, there is a collision with two prism-shaped objects and they are repainted to red.

A. Collision control in mixed load

Mixed loads consist of prism-shaped and cylinder-shaped objects. All objects are created from "Cylinder" or "Prism" class and obtain required attributes.

Collision algorithms and possible cases of prism-shaped and cylindrical-shaped objects orientations are shown in the Fig. 9.

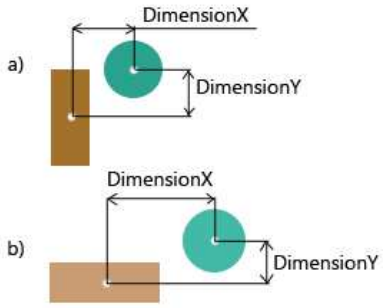

Fig. 9 Collision control between: a) 90° orientation of the prism and cylinder b) 0° orientation of the prism and cylinder

In the case of prisms and also in mixed load, variables *DimensionX* and *DimensionY* are calculated in the first step of the control. DimensionX is absolute value of difference between *x* coordinate of prism midpoint and *x* coordinate of cylinder midpoint. DimensionY is absolute value of difference between *y* coordinate of prism midpoint and *y* coordinate of cylinder midpoint.

$$\begin{aligned} DimensionX &= |BOXx - CYLINDERx| \\ DimensionY &= |BOXy - CYLINDERy| \end{aligned} \quad (14)$$

Conditions of not being in collision for those two variables are:

In case a)

$$\begin{aligned} &If\ DimensionY \le a/2 + diameter/2 \quad and \\ &DimensionX \ge b/2 + diameter/2 \end{aligned} \quad (15)$$

In case b)

$$\begin{aligned} &If\ DimensionY \le b/2 + diameter/2 \quad and \\ &DimensionX \ge a/2 + diameter/2 \end{aligned} \quad (16)$$

If those conditions are satisfied, then collision doesn´t exist. In the other case, there is a collision with two objects and those objects are repainted to red.

V. Virtual Reality Module

Coordinates of the end effector position do not provide excellent idea of the chosen solutions. Therefore, it was necessary to incorporate the graphic element to the program, which would indicate a solution in three – dimensional (3D) preview. Two – dimensional (2D) rendering would be sufficient only if we are analyzing one layer of the load. 3D models are better for multilayer solutions so it is necessary to use virtual reality interface. There are many interfaces that can provide rendering of the objects in space. We chose *Java3D graphic interface*, which includes well known *OpenGL GLUT library*. Libraries of Java3D interface were created in Java programming language. It is used mainly for modeling components, technical applications and technical product development.

It can provide rendering the object in space and allows the manipulation with it. It is also used in computer games and simulations. Before starting the program we have to install Java3D library. This library contains forms, shapes, functions, materials, scenes, movements, calculations, etc. The biggest advantage of Java3D interface is that we can have 3D model in the same window with GUI (Graphical User Interface) components like buttons, text boxes, labels, etc. We have used two *JPanel* components. The first contains command buttons with labels and the second is used as canvas for 3D preview.

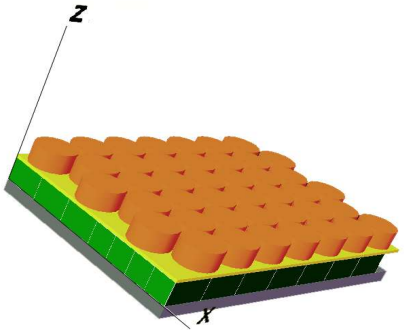

Fig. 10 Three dimensional preview of the load with the same objects in each layer

Dimensions of the object, dimensions of the pallet and coordinates with the rotations of objects are used as input data to the model. After drawing the solution into a virtual reality model, user can rotate and control the load by mouse and keyboard. This module can draw static or dynamic model of the load. Static model has no movements but we can control it with the mouse. Dynamic model is still turning around *z* axis.

VI. Using the Virtual Reality in Process of Mixed Load Planning

Virtual reality models are often used only for 3D preview of the real situation. But we have decided to use this strong tool also in the process of planning deployments. Mixed load consist of objects with different shapes and dimensions in the same layer of the pallet. It is very difficult to find the optimal deployment for it. So we had to divide storage area of the pallet (or layer area) on smaller areas. Those smaller areas are like small pallets so we can use deployment methods described in part 3 of this article for planning of new deployment there.

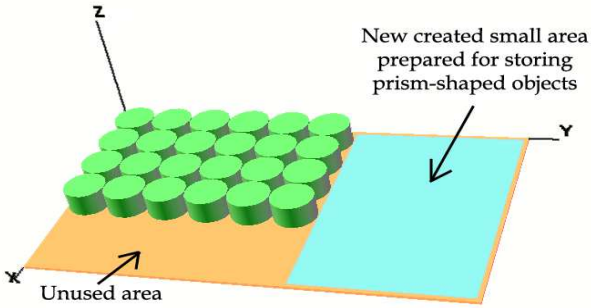

Fig. 11 Using the three dimensional preview for finding positions and dimensions for small areas on the pallet (layer)

Another problem is to find positions and dimensions for those small areas. For this purpose 3D editor was created that contains model of pallet and blue plane (it represents a new small area). We can manipulate with it and change its dimensions using the keys on the keyboard. So the output data of this module are coordinates of this small storage area and its dimensions (width and length). There is also a control system for finding collisions. There are only two types of collisions: *small area with another object on pallet* and "*flying" position of a small area* (if it is in the air).

VII. Robot Control Module

Inevitable function of this software is his communication with the SCARA robot. The control system knows several commands for controlling the movements of the robot. For example: *MOVE 100, 100, 90, 50* (go to coordinates 100, 100, 90, 50). Serial link is used for sending the individual commands and also for receiving the answers from the control system. This connection is made by the *USB port.* We created a simple window to control the robot from our computer. It contains a text line to send a simple commands and a text box to write the entire program for the robot. Another text box is used for writing the answers from the robot control system. If we send the entire program from the first text box to the control system, the error will occur because a buffer is full. The buffer of control system can handle only 50 lines of the program at once. So it is necessary to send the entire program line after line. To solve this problem we created *two synchronized threads.* The first thread sends individual commands to the control system of the robot (output) and the second thread reads replies from it (input). Thread output is inactive until the thread input does not catch the answer from control system of the robot. After catching the answer the next line of the program is sent by thread output and again waits for the response. The whole process will repeat until the entire program will be send line by line.

The second problem was the automatic generation of the commands for robot. So we created a template of commands to store only the one basic object. Variables are placed on places of coordinates to this template. These variables are automatically changed after each cycle of storage. Values of variables for catching the object from belt conveyor are from vision system module. Values of variables used for object positioning on the pallet are from deployment managing module. Each coordinate has a characteristic variable. If an error occurs in the control program of the robot, we can start it again from the last performed command. The last part of the window is a line that informs the connection status to robot control system via USB. If connection is ok, then the green sign CONNECTED is shown. If an error occurs, then the red sign DISCONNECTED is shown. Refreshing the connection can be done through the CONNECT button. All errors are shown in the INPUT text field with closer description of them. Additional function of this module is HISTORY of sent commands.

VIII. Obtaining Position and Class Information from Visual Signal

The vision system is developed under three signals from industrial cameras. The choice of three cameras was decided because of the complex possibility to analyze three dimensional constitution of the subject, even if for the product localization there is a need of at least one camera signal. For this solution was used the Basler camera with 33fps capture speed, SVGA resolution, 1394b firewire interface. They are connected to Pc via NI measurement card with PCI-e interface. The captured signal is preprocessed as we can see it in Fig. 12.

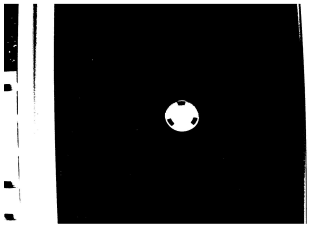

Fig. 12 Preprocessed camera signal

Here we can see the main view. From this preprocessed signal we extract the region of interest (ROI) as we can see in the Fig. 13.

Fig. 13 The region of interest

Our definition of ROI is based on the needs of the solution. By shortening the dimensions of this region we gain the shorter time of the processing but we can lose some important data. It is important to remember that any change of the ROI after the solution is made will cause the solution's malfunction.

The process of image preprocessing is demonstrated in Fig. 14, where the schema of preprocessing step is visualized. Camera signal is correctly fragmented into frames, every frame subjects to color adjusting resp. color conversion algorithm, where the best fitness of picture is obtained.

The color and grayscale picture is then removed to memory for next steps of vision analysis.

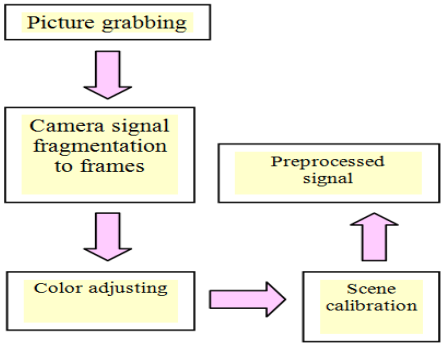

Fig. 14 Image preprocessing scheme

The processing of camera signal is executed by software, resp. software module, which creates the image preprocessing and vision analysis. Software implements Java jar library, where all of the steps are programmed, this was created because of easier way of developing the system. This principle is demonstrated in Fig. 15.

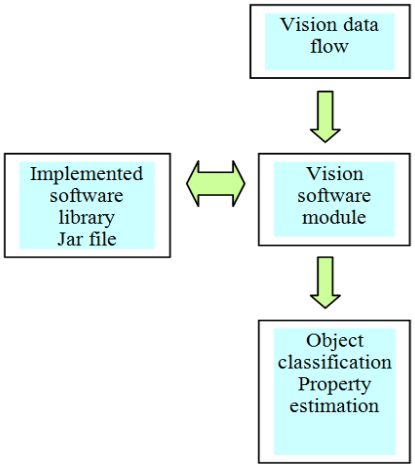

Fig. 15 Vision module scheme

The overall vision algorithm is demonstrated in Fig. 16. This algorithm classifies objects inputting into area of interest to several classes depending on the object geometry and of the color surface. Geometrical analysis is made using statistical methods created over closed regions of image after biasing and closing structures from grayscale image. After getting geometrical properties we obtain color information from color image and object coordinates. Obtained data are then stored to database or used by other software modules, such as robot program generation.

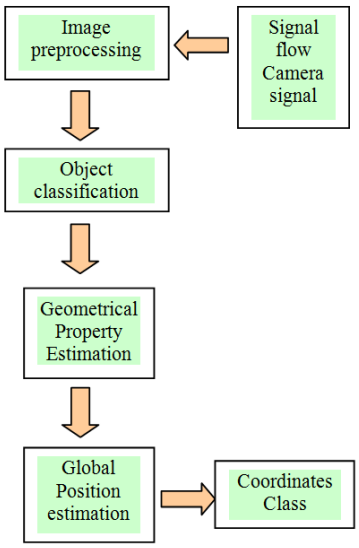

Fig. 16 Vision algorithm

IX. Conclusion

Knowledge of methods used for two-dimensional layout development is essential for creating three-dimensional and mixed loads. Many software products have implemented these methods in their algorithms. They use them for evaluation, calculation and creating optimal layout of objects on the pallet. If three – dimensional distribution is used then it is necessary to create a virtual reality model of pallet load. Thanks to this model we can analyze whole load and fill the empty spaces. In some cases it is necessary to divide pallets loading area into smaller areas. In each area we can use specific deployment method for identical objects. This solution is preferred for creation of mixed loads. This problem is solved in VEGA MŠ SR 1/0274/11 project.

References

[1] T. CORMEN, *Introduction to algorithms*. Massachusetts Institute of technology, 2009.
[2] M. HAJDUK, *Robotické bunky*. SjF Košice, 2008.
[3] A. HEINZE, *Optimisation of BMW Group Standardised Load Units via the Pallet Loading Problem*. Linkoping, 2006.
[4] T. KUHN, *Automatisierte Palettierung mit Mehrfachgreifern*. Universitat Hannover, Berlin, 1999.
[5] J. NELISSEN, *New Approaches to the Pallet Loading Problem*. Aachen, 1993.
[6] M. YANG, *Multi-layer palletization of multi-size prismes for 2D and 3D problems*. Montreal, 1993.
[7] D. KRAVEC, J. BAĎO, O. STAŠ, and M. TOLNAY, *Implementácia metód rozmiestňovania objektov na palete do nového softvérového produktu*. ROBTEP, Košice, 2011.
[8] P. Kovac, I. Mankova, M. Gostimirovic, M. Sekulic, B. Savkovic, *A review of machining monitoring systems*. Journal of Production Engineering, Vol. 14, No 1, pp 1-6, 2011.