ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
DOI: 10.17559/TV-20130412150016

SORTING SYSTEM WITH BUFFER STOCK VISUALISATION USING MATLAB GUI AND VRML

Martin Juhás, Bohuslava Juhásová

Original scientific paper
The paper describes the issue of an automated system of requirements processing visualisation. The sorting process of components is implemented by buffer stock using in this system. The system is realized by using the Matlab environment. The Matlab GUI elements are used for application design, which serves for system parameters definition and 2D process visualization. 3D visualization is processed by VRML, while system dynamics is controlled by Matlab commands. A series of experiments under various scenarios was performed. The experiments resulting in the system effectiveness analysis under various criteria based on defined system parameter changing.

Keywords: *buffer stock; Matlab GUI; requirements processing; system effectiveness; VRML*

Sustav razvrstavanja vizualizacijom robnih zaliha primjenom okruženja Matlab GUI i VRML

Izvorni znanstveni članak
U radu se opisuje problem vizualizacije automatiziranog sustava obrade potreba. Postupak razvrstavanja komponenti implementira se uporabom robne zalihe u tom sustavu. Sustav se ostvaruje uz Matlab okruženje. Matlab GUI elementi rabe se za dizajn aplikacije, koja služi za definiciju parametara sustava i vizualizaciju 2D postupka. Za obradu 3D vizualizacije primjenjuje se VRML, a dinamikom sustava se upravlja pomoću Matlab naredbi. Izvršen je niz eksperimenata s različitim scenarijima. Eksperimenti koji su rezultirali analizom učinkovitosti sustava primjenom različitih kriterija temeljili su se na promjeni parametara definiranog sustava.

Ključne riječi: *Matlab GUI; obrada potreba; robna zaliha; učinkovitost sustava; VRML*

1 Introduction

The aim of this work is to visualize an automated requirements handling system, where sorting of components is ensured by the use of components buffer stock. This system will be used as an experimental tool in education process in study programmes Applied Informatics and Automation in Industry and Automation and ICT Implementation in Processes of Institute of Applied Informatics, Automation and Mechatronics of Faculty of Materials Science and Technology in Trnava. Designed tool will serve for an effectiveness analysis of the proposed system under various criteria in case of defined system parameters changes.

Restrictive system requirements are:

- variable number of components
- variable size of input stack of components
- vertical buffer stock with variable number of cells
- operating manipulator with 3-axis motion
- the possibility of using up to three actuators
- variable requirements stack size
- the possibility of requirements batch processing
- the possibility of defined performance indicators monitoring
- the control application with
  - process 2D visualization
  - process 3D visualization.

2 System design

The system is implemented using Matlab environment [1 ÷ 4]. The Matrix Laboratory – Matlab is a powerful tool for technical computing, includes toolbox for 3D visualization and is mostly used for problems solving of laboratory exercises in our institute. In order to maximize the potential utilization of an implementation tool, the individual components of the system are implemented in the form of dynamic arrays (Fig. 1):

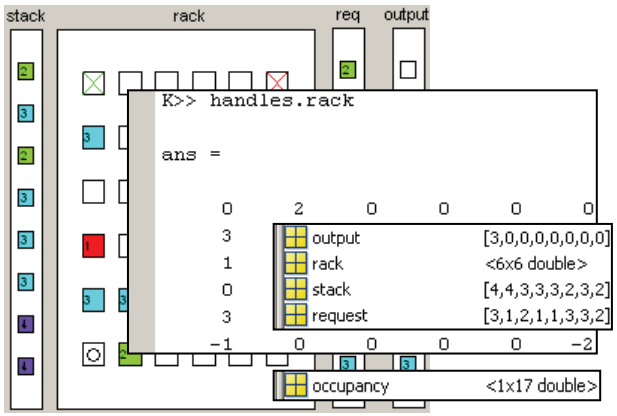

Figure 1 System data structure

- input tray of components: array [1 × Stack]
- the elements are generated randomly in the range of processed components in the system
- the tray is fully filled at the beginning of work
- after successful processing of current element, it is discarded from array and the end of the field is supplemented by a new randomly generated element
- buffer stock of components: array [Row × Col]
- the elements are complemented dynamically based on chosen requirements processing algorithm
- fixed output (unloading) cell is defined by the value –1
- fixed discarding cell is defined by the value –2
- requirements tray: array [1 × Req]
- the elements are generated randomly in the range of processed components in the system
- the tray is fully filled at the beginning of each cycle of requirements batch processing

– successfully processed requirements: array [1 × Out]
– the elements are replenished continuously based on chosen requirements processing algorithm
– storage occupation: dynamic size array [1 × …]
– an element is supplemented in each simulation step by the Eq. (1)

$$occupancy[i] = \frac{number\ of\ occupied\ cells}{number\ of\ available\ cells} \qquad a)$$

A structure type variable, which contains a copy of the dynamic arrays of input tray and requirements tray from the previous simulation run is used to ensure simulation reproducibility with modified system settings (Fig. 2).

```
K>> handles.old

ans =
        stack: {1x38 cell}
      request: {[2 3 3 1 3 3 2]
                [3 3 2 3 2 2 2]
                [2 2 3 3 1 2 2]}
```

Figure 2 History data of last 3–cycles simulation

3 System control algorithm

Following priority rules for incoming requests handling process were established:
- priority rule number 1: the requirement is handled by components input tray
- priority rule number 2: the requirement is handled by buffer stock.

The basic principle of the system operation which reflects specified priority rules for requirements handling is shown in Nassi-Shneiderman diagram form depicted in Fig. 3.

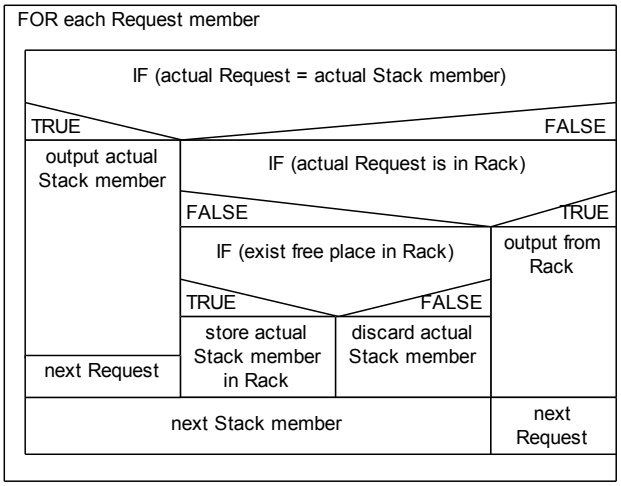

Figure 3 Request processing algorithm

Two methods for searching of free cell and requested component in buffer stock are used:
– the FIND method direct implemented in Matlab (marked as DEFAULT), which scans the target array (tray) always from the beginning
– the custom method (marked as RING) based on calculating the distance between defined elements and then selecting the element with minimum distance.

The requirement execution is realized either through predefined storage cell, or alternatively through any unused buffer stock cell (identified by the chosen search algorithm).

To avoid algorithm deadlock one of following two options is used:
– one buffer stock cell is permanently defined for discarding only
– component discarding is executed through any unused buffer stock cell, while next component storing is possible only in case if at least one cell remains unoccupied after storing process in buffer stock.

4 System visualization

4.1 Matlab GUI application

The system visualization is realized by Matlab tool GUIDE in the application form [1, 2, 5, 7], whose interface is shown in Fig. 4.

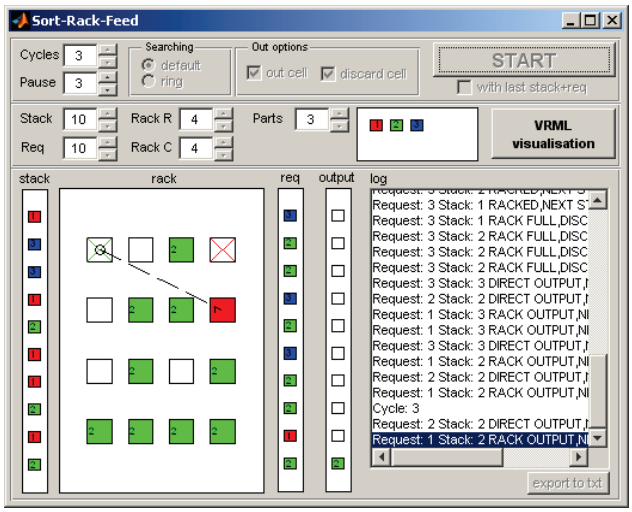

Figure 4 System application GUI

The application allows defining:
– the number of elements of the components tray (Stack)
– the individual elements are differentiated by colour based on the HSV colour map, as well as by numerical designation of the component type
– the number of elements of the requirements tray (Req)
– buffer stock dimensions (Rack R – rows, Rack C – columns)
– the number of types of components in the system (Parts)
– the number of requirements batch processing repetitions (Cycles)
– an empty cell search method for component storage into buffer stock, respectively stored component marked for export search method (Searching – default/ring)

– the existence of a cell designed for direct output from input tray (out cell)
– the existence of a cell designed for input tray element discard (discard cell)
– the simulation speed (Pause).

The application also allows the simulation experiment repeating with identically generated inputs tray and requirements try for:
– type of search algorithm changing
– the existence of output, respectively discard cells changing.

The simulation repeating is not possible in case of input tray, requirements tray, buffer stock size changing, or number of component types in the system changing.

Statistical information observed during the simulation:
– the simulation duration
– the manipulator trajectory length, which is given in relative units for elements horizontal and vertical spacing equal to 1
– average buffer stock occupancy
– the number and percentage of discarded elements.

The simulation progress and the resulting statistical information are stored in text form in the list box (log). For example, for 1 simulation cycle with 2 element types, ring–type of search, 2 × 2 stock size with discard cell, without default output cell, two–pieces input tray and three-element requirements tray, the resulting log is shown in Fig. 5. After the simulation, the simulation log can be exported into text file.

```
START: 09:27:46
Cycle: 1
Request: 2 Stack: 1 RACKED,NEXT STACK
Request: 2 Stack: 2 DIRECT OUTPUT,NEXT REQ
Request: 1 Stack: 1 DIRECT OUTPUT,NEXT REQ
Request: 1 Stack: 2 RACK OUTPUT,NEXT REQ
STOP: 09:27:46
Elapsed time: 0.45678 s
Trajectory: 2
Rack occupancy: 25.00 %
Discarded: 0 of 3 (0.00 %)
```

Figure 5 Simulation log example

4.2 VRML virtual world

3D visualization of the system is implemented using VRML [6] combined with tool Matlab (Fig. 9) [8]. The virtual reality model consists of a combination of several structures.

4.2.1 Statically defined structure of system

This structure is shown in Fig. 6 and consists of:
• system base
• buffer stock frame with the first horizontal partition defining a base of the storage
• the tray of:
– components types
– requirements (created as a clone of the component types tray using USE)
– output (created as a clone of the component types tray using USE)
• manipulator with possibility of moving in the *x* and *y* axes, containing:
– components input tray
– handling element providing movement along the *z* axis
▪ point light source
▪ 2 defined viewpoints
▪ virtual world background.

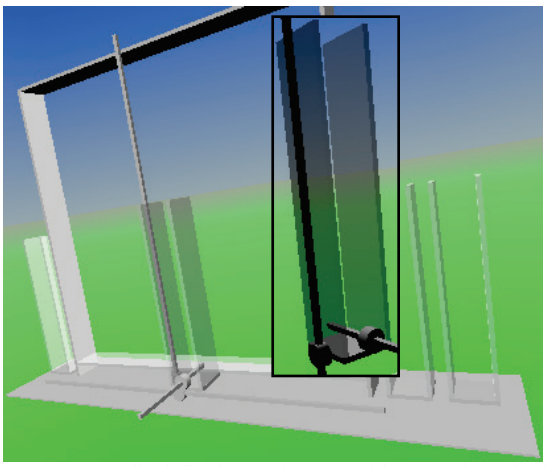

Figure 6 Statically defined part of system with manipulator detail

4.2.2 Dynamically generated structure of system

Dynamically generated structure is based on the external prototypes using Matlab commands for changing the properties **Translation** and **material. Diffuse Color** (Fig. 7) consisting of:
– components
▪ samples in the component types tray
▪ required in the requirements tray
▪ input in the input tray
▪ successfully processed in the output tray
▪ stored in buffer stock
– vertical partitions indicating the number of storage columns
– horizontal partition indicating the number of storage rows
– export and discard cell marking.

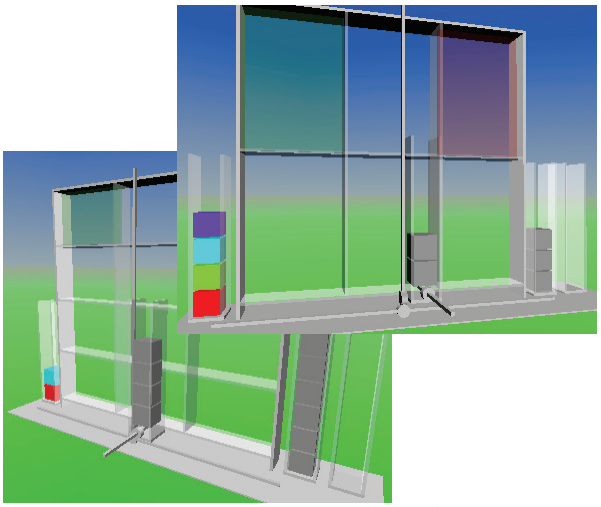

Figure 7 Dynamically generated part of system

4.2.3 Process animation

Animation of sorting process is created by:
- three independent objects **Timer Sensor** whose activity is controlled by Matlab commands changing their property **enabled**
- four dependent objects **Position Interpolator** whose definition of key values of property **Key Value** is performed by Matlab commands based on an ongoing simulation
- set of eight tightly defined objects **Route**.

Dynamic scene generation is based on the principle of static scene basic coordinate system shown in Fig. 8.

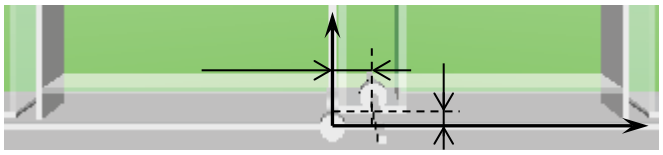

Figure 8 The basic coordinate system of static scene

The movement of the manipulator is due to dynamically created scene recalculated based on the Eq. (2)

$$\begin{aligned} &cell\_width = rack\_width / rack\_c; \\ &cell\_height = rack\_height / rack\_r; \\ &x\_target = -rack\_width + (cell\_width/2) \\ &\quad +((col-1)*cell\_width) - x\_diff; \\ &y\_target = y\_diff + (row-1)*cell\_height; \end{aligned} \qquad b)$$

where:
- rack_width is the overall width of the rack (in relative units of virtual scene)
- rack_height is the overall height of the rack (in relative units of virtual scene)
- rack_c is the number of rack columns
- rack_r is the number of rack rows
- col, row are the relative position of the storage cell (row and column indexes of the cell).

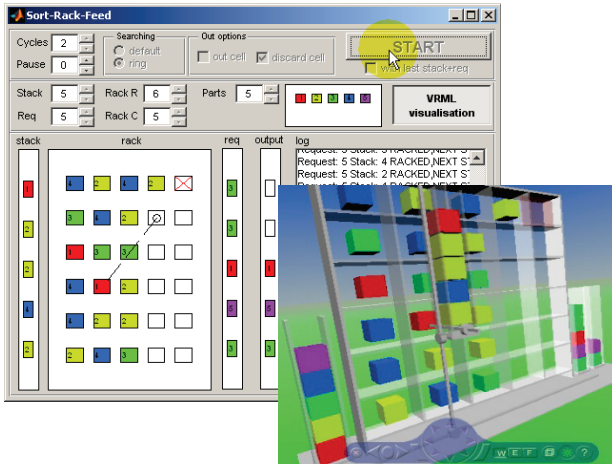

Figure 9 Process visualization "in action"

5 Scenarios of experiments

A basic set of experiments was performed on created model under the following scenarios.

5.1 Scenario No. 1

Table 1 System parameters for Scenario No. 1

cycles	rack	stack	req	parts	search
5	5×5	10	10	5	default

This experiment was performed 10 times to obtain the basic characteristics of the chosen system configuration, which makes it possible to derive scenarios for further experiments. Basic monitored characteristics, which are the basis for the next group of experiments, are in this case:
- the percentage of storage occupancy
- the percentage of discarded elements.

Additional characteristics are:
- the duration of the simulation
- path (relative) conducted by manipulator.

5.2 Scenario No. 2

The basic setting of model is the same as in Scenario No. 1. The starting point is the simulation experiment, where the basic characteristic corresponds to the average value of previous experiment characteristics. The impact of combination of following system parameters changing is monitored:
- the empty cell searching algorithm while storing it into rack, respectively while identifying compliant element in storage
- the existence of the output cell
- the existence of the discarding cell.

The primary monitored features in this case are:
- the duration of the simulation
- path (relative) conducted by manipulator.

This experiment was performed 3 times for each combination of varying system parameters.

6 Experiments results

Table 2 Experiments results of Scenario No. 1

	Time / s	Trajectory	Occupancy / %	Discarded
1.	24,4857	201,9163	66,43	11 of 73 (15,07 %)
2.	26,5636	247,8316	80,06	8 of 78 (10,26 %)
3.	30,8506	197,4774	82,43	28 of 99 (28,28 %)
4.	21,1238	184,9176	54,06	1 of 67 (1,49 %)
5.	18,7934	182,0449	52,59	0 of 56 (0,00 %)
6.	23,8816	219,4995	55,51	12 of 82 (14,63 %)
7.	31,5705	204,2403	84,10	37 of 108 (34,26 %)
8.	39,4689	261,2089	79,18	64 of 137 (46,72 %)
9.	24,1346	211,0167	58,49	15 of 78 (19,23 %)
10.	31,6616	207,6093	78,36	41 of 101 (40,59 %)
Avg	27,25343	211,77625	69,12	21,053%

The results obtained suggest that, at given configuration of the system, the average percentage store occupancy is almost 70 % and the percentage of the discarded elements is about 20 %.

Table 3 Experiments results of Scenario No. 2

		+o+d	+o-d	-o+d	-o-d
def	*T*	25,3097	25,0897	25,206	24,9497
	Tr	212,3681	183,4912	165,6488	165,6488
	Oc	69,77	69,46	69,46	66,68
	D	17 of 80 (21,25 %)	16 of 80 (20,00 %)	16 of 80 (20,00 %)	16 of 80 (20,00 %)
ring	*T*	25,2986	25,1388	25,4903	24,9466
	Tr	162,7632	131,5272	89,2001	85,1356
	Oc	69,77	69,46	69,46	66,68
	D	17 of 80 (21,25 %)	16 of 80 (20,00 %)	16 of 80 (20,00 %)	16 of 80 (20,00 %)

Note: +o/–o − output cell defined/undefined; +d/–d − discard cell defined/undefined; def / ring − default / ring searching algorithm; *T* − time (s); *Tr* − trajectory; *Oc* − occupancy (%); *D* − number of discarded parts.

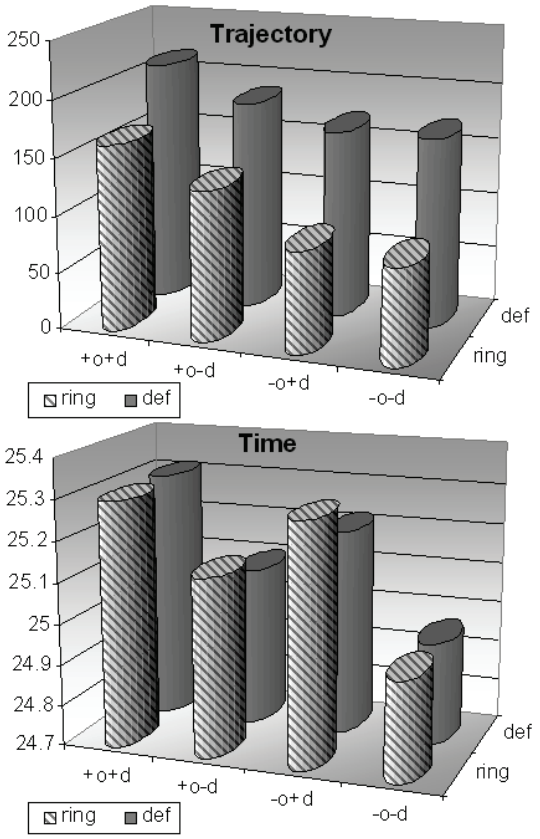

Figure 10 Scenario No. 2 results

From Tab. 3 and Fig. 10 it is clear that in terms of time the search algorithm selection has not a significant impact on system efficiency. In terms of existence of fixed output or discarding cells it appears to be the most effective option, where the output or discarding cell is not defined, while item output and item eliminating are realized through an unused cell. This advantage is negatively balanced by necessity to ensure an additional elements flow control for each cell (successful exit / rejection). In terms of the trajectory length the RING search algorithm is more convenient (which has no significantly negative aspect in the time domain). Similarly as in the previous case, the most advantageous option is when no output or discarding cell is defined.

7 Conclusion

A process of visualization and analysis of automated system of requirements processing, in which components of input tray sorting is implemented by buffer stock, is presented in this article. As a simulation tool the Matlab software was used. System visualization was processed using the GUI directly in Matlab and 3D representation in the VRML language linked to the GUI. A basic group of experiments has been performed that serves as an example of using the created system to simulate different scenarios of different system configurations monitoring.

One of the main solution benefits is possibility of using it as an experimental tool in education process.

Possible directions for the future of the project:

- providing of additional required cell selection algorithms based on various priority rules
- the use of simulation optimization to determine the optimal parameters of the system according to defined criteria
- physical realization of the system using automatic control hardware such as LEGO Mindstorms or PLC system in cooperation with Matlab
- extension of more concurrent parallel storages.

Acknowledgements

This publication was written with financial support of the VEGA agency in the frame of the project 1/0463/13 "Study of flexible mechatronics system variable parameters influence on its control".

8 References

[1] Bansal, R. K.; Goel, A.; Sharma, M. K. MATLAB and Its Applications in Engineering. Dorling Kindersley (India) Pvt Ltd, 2009.

[2] Hunt, B. R.; Lipsman, R. L.; Rosenberg, J. M.; Coombes, K. R.; Osborn, J. E.; Stuck, G. J. A Guide to MATLAB: For Beginners and Experienced Users. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511791284

[3] Kozák, Š.; Kajan, S. MATLAB – SIMULINK I. Vydavateľstvo STU, Bratislava, 1999.

[4] Juhás, M.; Juhásová, B. Basics of automated control. Exercises manual. // AlumniPress, Trnava, 2012.

[5] Lent, C. S. Learning to Program with MATLAB: Building GUI Tools. Wiley Global Education, 2013.

[6] Žára, J. VRML 97 – Laskavý průvodce virtuálními světy. Computer Press, Praha, 1999.

[7] MathWorks. Matlab GUI. URL: www.mathworks.com/discovery/matlab-gui.html (19.08.2015)

[8] MathWorks. Matlab Interaction. URL: www.mathworks.com/help/sl3d/matlab-interaction.html (19.08.2015)

Authors' addresses

Martin Juhás, Assist. Prof. Dr.
Institute of Applied Informatics, Automation and Mechatronics
Faculty of Materials Science and Technology STU in Bratislava
Hajdóczyho 1, 917 24 Trnava, Slovakia
martin_juhas@stuba.sk

Bohuslava Juhásová, Assist. Prof. Dr.
Institute of Applied Informatics, Automation and Mechatronics
Faculty of Materials Science and Technology STU in Bratislava
Hajdóczyho 1, 917 24 Trnava, Slovakia
bohuslava.juhasova@stuba.sk