EG UK Theory and Practice of Computer Graphics (2006)
M. McDerby, L. Lever (Editors)

Visual Analysis of Packing Process for 3D Container

Y. Yue, M. Middleton and J. Liu

University of Luton, UK

Abstract

The cutting and packing problem has been encountered in many industrial sectors, and become a research focus in operations research. Because of its nature, it is a commendable goal to visualise the process of cutting and packing for analysis and validation of the algorithms. This is even more desirable when working in a 3-dimensional environment. There have been some visualisation packages for which the working algorithms are hidden behind the screen. Furthermore, their effectiveness and flexibility are limited in some sense. This research presents a visual analysis tool for 3-dimensional container loading. A new loading and display algorithm is devised to suit requirements for container loading. Test results are given with recommendations for further effort.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Display Algorithms

1. Introduction

The cutting and packing problem has plagued many industries over the years. From warehouses to manufacturing plants, the ability to optimise a packing or cutting process can mean the difference between profit and bankruptcy. Of the many analogies of this problem, perhaps the easiest to follow is that of packing a car boot. Assuming we have to transport many items of varying shapes and sizes using a car, the order and placement of the items when we load them into the car boot can greatly affect the total number of journeys required to transport the items. The ability to plan and visualise the loading process could save time and effort and present an optimal loading process, resulting in fewer journeys. The same process could apply to a warehouse, which is attempting to fill containers with boxes of varying sizes. The placement of and order in which the boxes are loaded can greatly affect the number of containers required to ship an order.

This paper provides a brief description of the cutting and packing problem and investigates the current graphical display options available. A visualisation program is developed to depict and aid the analysis of 3D container loading. The implementation and test results are presented along with concluding statements.

2. Visualisation of the cutting and packing process

This section provides a review covering some of the visualisation options available for the cutting and packing problem. There are commercially available applications which can be used to help optimise the process of loading containers. For example, in the shipping industry, 3D Load Packer [Ast06] and Cube-IQ [Mag06] are two of the applications which are discussed in further detail below.

2.1 3D Load Packer

3D Load Packer (3DLP) is a simple application, created by Astrokettle, which offers a number of functions to aid the packing process of shipping containers. The software itself allows the user to specify a single (or multiple) container(s) and a list of one or more box-shaped objects to pack into the container(s). Upon starting 3DLP, the user is presented with a somewhat colourful interface (Figure 1). It is split into three sections: container selection, box selection and display of information about the current problem, offering the options for solution. A number of preset containers and boxes exist in a built-in database, which can be extended with the addition of custom containers and boxes.

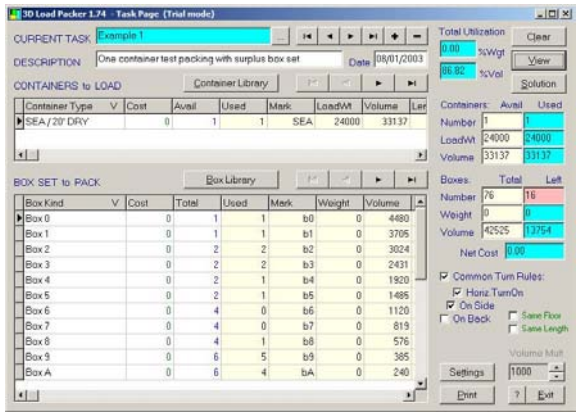


Figure 1: *3D Load Packer's packed container view*

Once the container and boxes have been selected, the problem can be 'solved' and the optimal packing process is calculated according to its built-in algorithm. Following loading, the fully packed container can be viewed on screen (Figure 2).

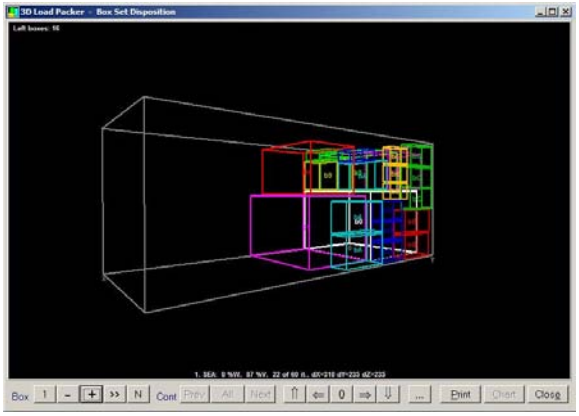


Figure 2: *3D Load Packer's packed container view*

The packed container view offers by default, a wireframe 3-dimensional representation of the container, containing the boxes in their positions calculated by the packing algorithm. Users are offered a number of customisation options, giving many flexible types of view; the container can be rotated and scaled to the user's desired position and size, colours can be cycled to assist in the identification of boxes and boxes can be labelled in position to aid further the identification process. The user can also select the number of boxes to be packed – boxes can be removed and added as per their load order, giving the user the perceived ability of being able to "watch" the boxes being loaded in their optimal positions. This process can also be automated, requiring no user intervention between steps.

However, its main display is rather confusing. When displaying a completely filled container, the mishmash of wireframe boxes can make the identification of boxes difficult – and the box labels, also displayed in an effort to help identification, appear to have the opposite effect, making the display a mix of text and lines and fairly difficult to look at. It is in this area that perhaps 3DLP fails to provide the desired visualisation function – the packing algorithm itself appears to perform remarkably but, when viewing results, the user is left feeling somewhat confused and unsure as to what is being shown. Another frustrating part of this application is that boxes must be added to a container one at a time, a process which can take quite some time given the programs non-standard interface. It can only cope with box-shaped objects.

2.2 Cube-IQ

Cube-IQ, developed by MagicLogic, is a more advanced application than 3DLP, offering many functions to adjust the final visualisation and provide the necessary information. The main program interface (Figure 3) itself is similar to a Microsoft Office application, and most sections are instantly recognizable. A small preview window is provided to show the progress of adding boxes to a container – this small preview can then be expanded to provide a more comprehensive view, covered in detail below. This application also offers the ability to pack cylindrical and 'sofa' shaped objects – obviously aimed at the shipping industry and the various awkwardly shaped packages that may need to be shipped.

The main interface is intuitive and clearly labelled, allowing for the easy set up of one or more containers, each containing one or more boxes. Box properties, such as dimensions and colour, can easily be modified, and boxes can be added in varying quantities. An optimise button provides access to the packing algorithm, which checks the various packing options and returns the most efficient packing order and placement. This result can then be viewed as a packing plan (Figure 4).

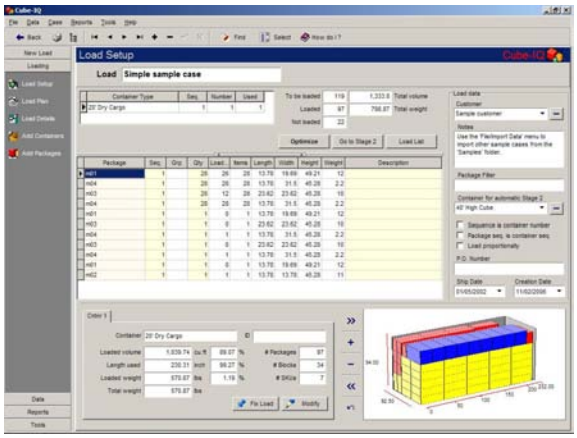


Figure 3: *Cube-IQ's main interface*

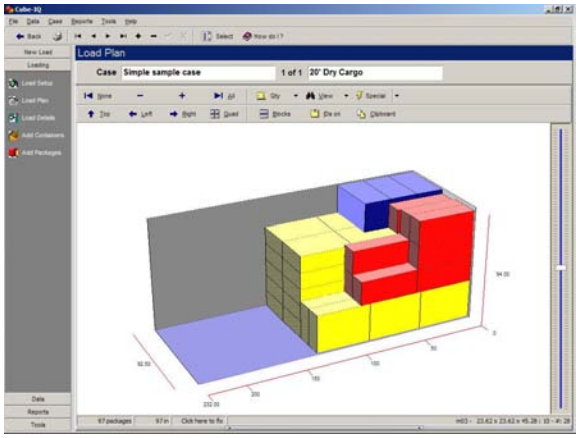


Figure 4: *Cube-IQ's packed container view*

The default packing plan interface provides a colourful, solid, 3-dimensional representation of the packed container. Boxes can be added and removed as per their load order and, by moving the mouse pointer over a box, a small label will appear identifying the box. Other view options include the ability to jump to various preset views – orthographic left, front and top views can be selected, as can a four-way view looking at the container from all four corners (Figure 5). Another interesting feature is the explode option, allowing the boxes to explode beyond the container, giving an easier impression of how the boxes will interact with each other in the final packed container (Figure 6).

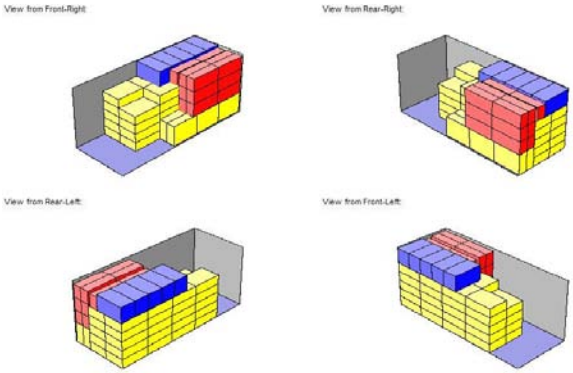


Figure 5: *Cube-IQ's quad view*

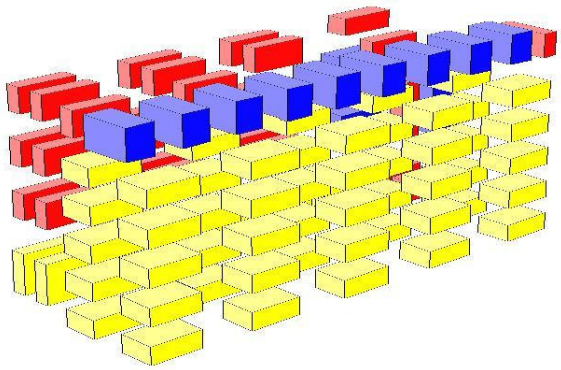


Figure 6: *Cube-IQ's explode view*

Further options include the ability to view the packed container in wireframe (as per 3DLP), and a 'ghost' option which draws the packed container using transparent materials, giving the ability to see through and behind boxes – the results are somewhat disappointing however, and perhaps as confusing as the wireframe view, if not more so.

While Cube-IQ provides a graphically-impressive display of the packed container, access to the individual box details (such as their position within the container, or specific load order number) are hidden away separate from the graphical view. This forced switching between windows to correlate information between the box list and graphical view is rather tedious, and something which should not have been overlooked when the application was being developed.

2.3 Summary of review

Both 3DLP and Cube-IQ offer similar functions in differing fashions. 3DLP's approach is somewhat simplistic, but this simplicity leads to a smoother and quicker packing process, resulting in quicker displays of the packed container. Cube-IQ, conversely, provides extra functions and usability, at the cost of speed. Both applications come with hefty price tags, and the underlying algorithms of the software, which perform the actual packing optimisation, are closely guarded secrets of the developers.

3. The cutting and packing problem

The cutting and packing problem affects two separate areas: cutting applications (such as the manufacturing industry, where the optimisation of the use of materials is imperative) and packing applications (such as the shipping industry, where the use of space is equally imperative). It is also possible to separate the cutting and packing problem into two domains: 2-dimensional and 3-dimensional packing problems. This research examines visualisation in the 3-dimensional domain.

3.1 3D packing

We shall look at the packing of cuboid shaped objects into cuboid shaped containers. In the real world, awkwardly-shaped items, such as cylinders or spheres, may need to be packed into equally awkwardly-shaped containers. However, these types of packing problems are too complicated for this research in terms of the packing and optimisation algorithms, and we will only discuss the packing of cuboid shaped objects, i.e. boxes.

When packing a box into a container, a number of factors must be taken into account. Perhaps the most obvious factor to consider is whether the box will fit into the container – that is, the box's dimensions can be less than, but no more than the container's dimensions. Depending on the box dimensions, it may be possible to arbitrarily rotate the box, in order to make the box fit. This process could be seen as mapping the box dimensions to the different dimensions of the container. A box may be too tall to pack vertically into a container, but by placing the box sideways (mapping the box's height to the containers length), packing may become possible.

A restricting factor of packing a container is the box weight. Certain containers may not be suitable for heavy boxes, and conversely certain containers may not be suitable for light boxes. Containers may also incur a cost for use and, of course, multiple containers will increase these costs further.

Another, often overlooked, factor which can affect the packing of a container is the way in which it must be packed. Generally, large shipping containers are packed from one end – boxes are loaded in one end and placed at the opposite end of the container. However, it is more than possible that a container may need to be packed from the top down, or have some other specific loading requirements.

3.2 Packing strategies

There are various strategies that can be undertaken when packing a container. Possibly the most obvious packing strategy is the first-fit strategy. This strategy will rank firstly all boxes in decreasing order of volume, or by edge dimension of the boxes. It will then pack boxes into a container until the next box will not fit [GR81, JCH00]. At this point the container is sealed and a new container is used to continue the packing. This clearly could lead to a large amount of wasted space – the box selected to pack which does not fit may be a large box. It is more than

possible that there may be smaller box(es) that will fit into the remaining space.

Another strategy can be thought of as a best-fit strategy [Pis02]. Boxes are also packed in their order of edge dimension – the largest possible box is packed first, followed by the largest box that will fit into the remaining space, and so on. The best-fit strategy is always to find the best-fit box to be packed into the next remaining space wherever possible. Depending on the boxes to be packed, this strategy can lead to better results than the first-fit strategy.

4. Development of the visualisation program

As discussed previously, visualisation software is available to help display and analyse the process of packing. However, these software packages invariably come with high price tags and working algorithms hidden behind the screen, and for this reason, it is proposed that a new software package is created, using a packing algorithm devised by the authors. The development of this software package is discussed below.

4.1 Requirements

The software itself will provide an easy interface to allow the selection of containers and boxes to pack, with the ability to view a 3-dimensional representation for the process of packing the container. The 3-dimensional view should provide the user with the necessary information required to identify boxes and their loaded positions within the container.

The visualisation should provide access to a number of view options, giving the ability to view the packed container from any angle and zoom in/out as required. The ability to view the packed boxes and their respective order should also be given, to aid in the identification of the load order of boxes.

4.2 The packing algorithm

The main packing algorithm itself requires some development. For the discussion of development, we consider the packing of one container, though applying the algorithm to more than one container should not present a problem. It is based upon a best-fit strategy.

A given set of n boxes is packed into containers, having length, width and height (c^l, c^w, c^h). Each box is characterised by its dimension properties (b_i^l, b_i^w, b_i^h), $i = 1, 2, \ldots, n$. All boxes are required to meet the following constraints:

1) The dimensions of boxes must satisfy: $\max\{(b_i^l, b_i^w, b_i^h)\} \leq \max\{c^l, c^w, c^h\}$, $i = 1, 2, \ldots, n$.
2) Boxes are packed into a container orthogonally that provides lateral support to the boxes without overlapping.
3) Overhang is allowed provided the stability of the box is maintained, i.e. the bottom surface of the box is sufficiently contained within the top surface of the box below it.
4) Boxes can only be packed along one edge of the container, from the far side to the near side and from the bottom to the top.
5) All boxes can rotate in the three orientations.

The objective is to obtain the maximum volume utilisation of the container. The algorithm begins by gathering the properties of the container which is to be packed - the dimensions of the packing space with packing boxes, not the container dimensions. This packing space can be conceived as a large, single space. Once these properties have been recorded, the algorithm can begin selecting boxes to pack.

The boxes to be packed into the container, held in a box-list, in the decreasing order of their volumes, have several properties: length, width, height, weight and cost. For the time being, the algorithm will be concerned only with the dimension properties of each single box (b_i^l, b_i^w, b_i^h). The algorithm searches through the box-list for the largest box which will fit into the given space (remembering to check the various rotations of the box). The largest box is packed in the direction of its largest section surface into the given space. Once the largest box has been identified, it is allocated a position within the container (in the case of the first box, at the far corner of the container).

Once a box is packed into the container, the space available is reduced. It can be conceived that the large, single space of the container has been split into two sections. The first section occupies the space that contains the packed box, having the length of the box, called the current space (b_1^l, c^w, c^h) and, the other represents the remainder of the container space ((c^l - b_1^l), c^w, c^h) (Figure 7). If any boxes of the same height as, and a similar length to the first packed box, are found from the box-list, and packed into the current space on the container floor; then, the current space is further split into two sections: the upper remaining space (b_1^l, b^w, (c^h-b_1^h)), b^w being the total length of boxes packed, and the front remaining space (b_1^l, (c^w - b^w), c^h) (Figure 8a). Otherwise, the current space is directly split into the upper remaining space (b_1^l, b_1^w, (c^h-b_1^h)) and the front remaining space (b_1^l, (c^w – b_1^w), c^h) (Figure 8b). The remaining spaces are considered in turn, in the order of the front, the upper and the right spaces.

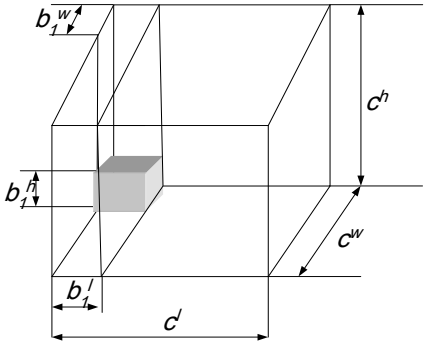


Figure 7: *The space split of the container*

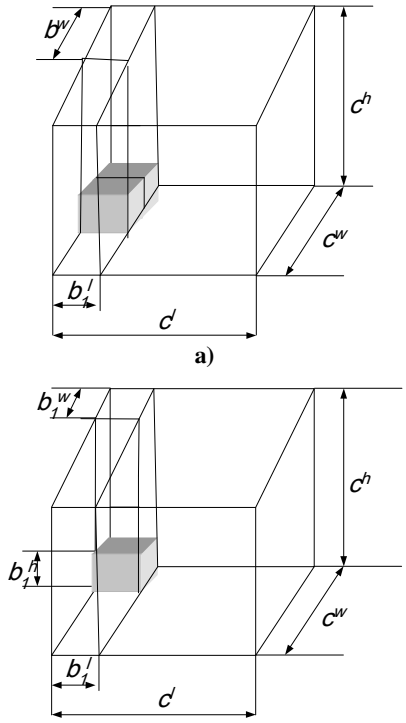


Figure 8: *The remaining spaces after boxes packed*

This procedure of packing and splitting is repeated iteratively until no further boxes can be packed into any remaining space.

During the packing procedure, boxes which cannot fit into the container should be ignored, or added to an 'unpackable' list, to be dealt with later on. Some pseudo code for the proposed packing algorithm is shown Figure 9.

4.3 The implementation

The program is implemented using the C# language, and the TAO .NET framework [TAO06] to give access to OpenGL [Ope06] functions. The main form provides options to select containers and boxes to pack, including the ability to adjust a box's dimensions and colour. Boxes can be added in multiple quantities, as required. Once the container and boxes have been selected, the program will solve the packing problem while displaying the process. To allow visual analysis of the packing process, the program has incorporated the following features. The packed container view displays a white wireframe box to represent the container with a black background. Boxes are represented by cubes of varying colour, which is scaled to match the relevant box dimensions. The main display can be rotated and scaled to give a zoom effect, using the mouse. Accompanying the display is a list of the currently-packed boxes. Each box can be highlighted by removing the surrounding boxes, and boxes can be removed and added to the container, as per the load order. This process can also be automated, viewing the container as it gets packed over time. All of the information remains separate from the container display, keeping a consistent and clean feel to the visualisation.

5. Testing of the program

The software has been created and developed as specified above. The main application interface (Figure 10) provides the container and box options necessary to specify a packing problem. Multiple containers can be added to a problem, and multiple boxes can be added to any container. The container options (Figure 11) allow the user to select either a pre-defined container, or enter their own details, specifying the dimensions.

```
set_initial_container_list                    //Initialise with target container

while container has usable space
       get_container_to_fill                  //Get next container from list
       for each box in boxlist
              find best-fit box               //Find best-fit box for container

       pack_box

       partition_remaining_space              //Split container into two sections
       add_containers_to_list                 //Add sections to container list

       for each box in box_list
              find best-fit boxes with the same height
              pack_box
              partition_remaining_space       //Split container into two sections again
```

Figure 9: *Pseudo code for the proposed packing algorithm*

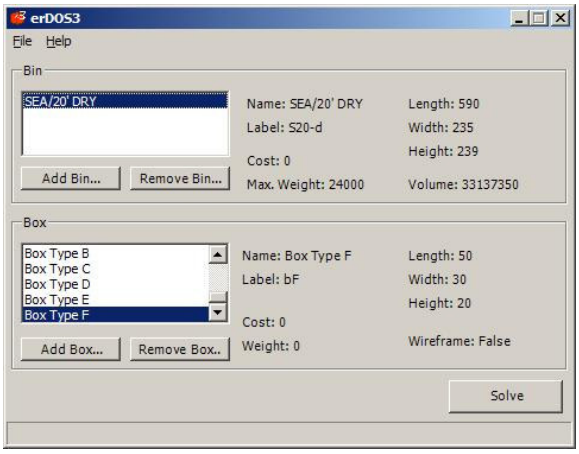


Figure 10: *The main interface*

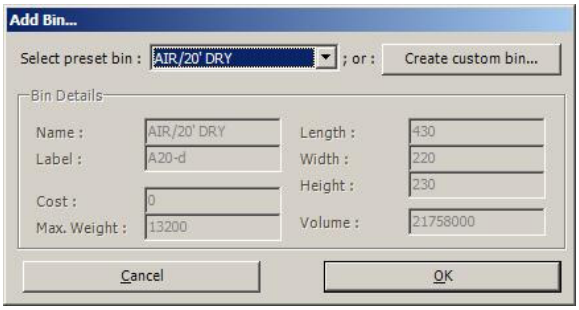


Figure 11: *Container specification*

Box options (Figure 12) include the ability to select a pre-defined box or enter a custom box. The colour of the box can also be adjusted if necessary (colours are randomly generated otherwise). Finally, the packed container view (Figure 13) shows, by default, a side-on view of the packed container. This view can be rotated and zoomed using the mouse, giving the ability to view the container from any angle.

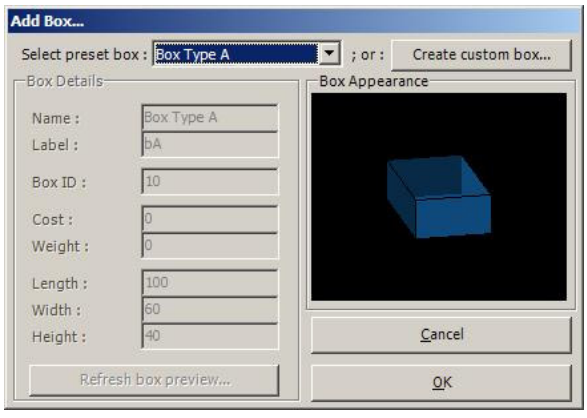


Figure 12: *Box options*

The view interface itself is very intuitive – rotation of the container appears fluid and natural. The ability to view from any angle leads to some practical views of a packed container (Figure 14). Individual boxes can be highlighted by selecting the relevant box from the list on the right – this option actually switches all but the selected box to wireframe mode, making easy identification of the selected box (Figure 15). By using the box control buttons, boxes can be added to the container as they would be loaded, giving the impression of loading the container over time (Figures 16-18).

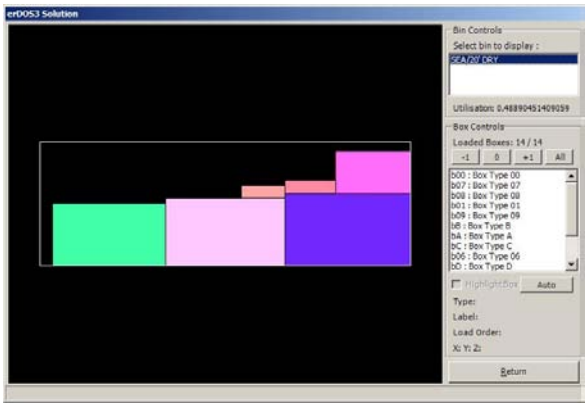


Figure 13: *Default packed container view*

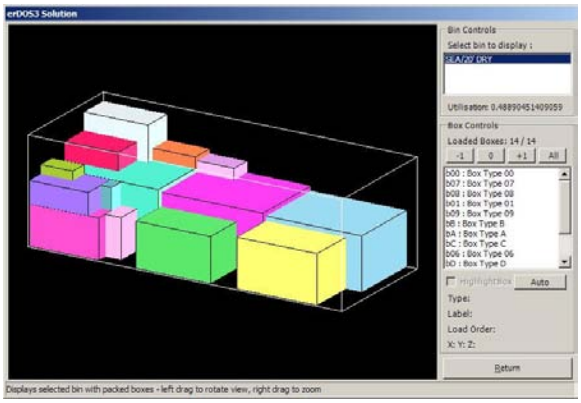


Figure 14: *Rotated packed container view*

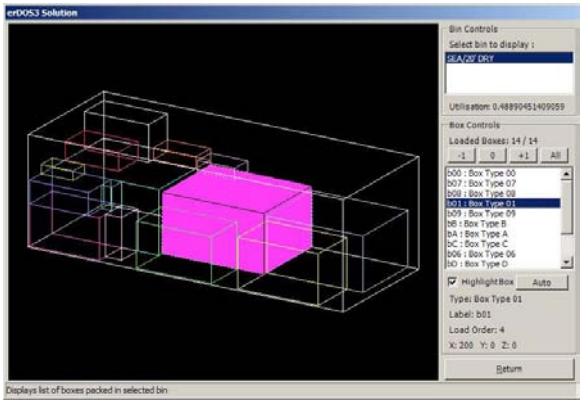
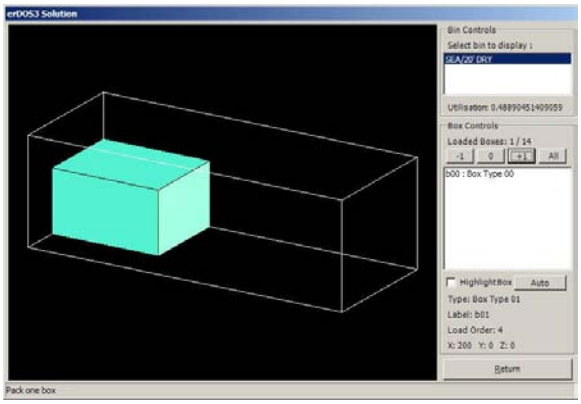
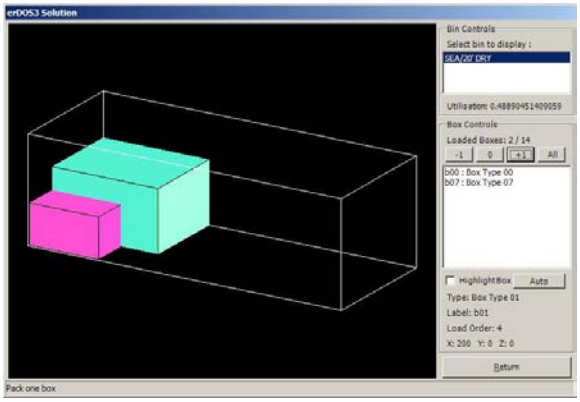
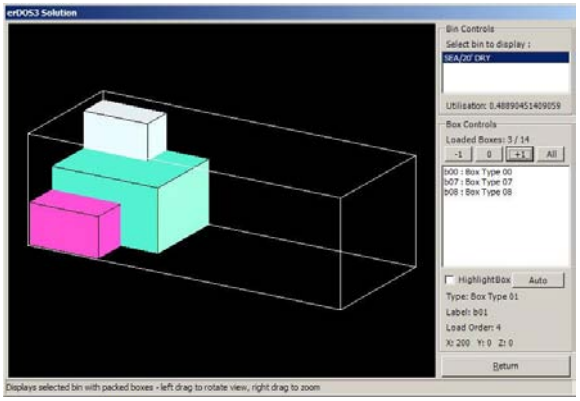


Figure 15: *Highlighted box*

The algorithm itself performs admirably given its simplicity with acceptable results (Figure 19) and in some case with very encouraging results (Figure 20).







Figures 16-18: *Adding boxes by load order*

6. Concluding remarks

The software implemented from the algorithm is a successful venture into the cutting and packing problem. Although the terminology refers to packing a container, the same results could easily apply to a cutting process. Although successful, the results are not entirely optimal, and the algorithm will require attention in order to achieve optimisation.

The display of the packed container was an overall success. The display is not cluttered with box-related information, and individual boxes can easily be highlighted as needed. The software gives an equally impressive impression as the 3DLP and Cube-IQ programs mentioned previously, without the clutter or confusing options.

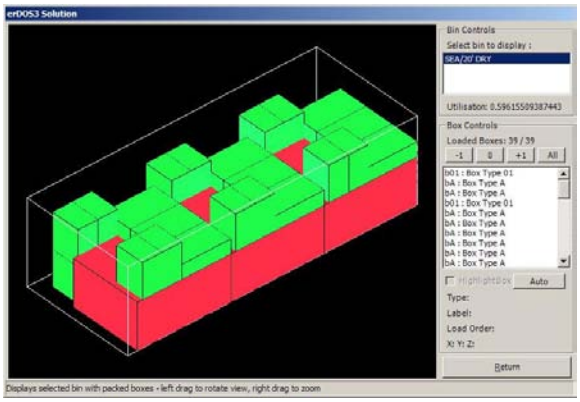


Figure 19: *Smaller problems give feasible results*

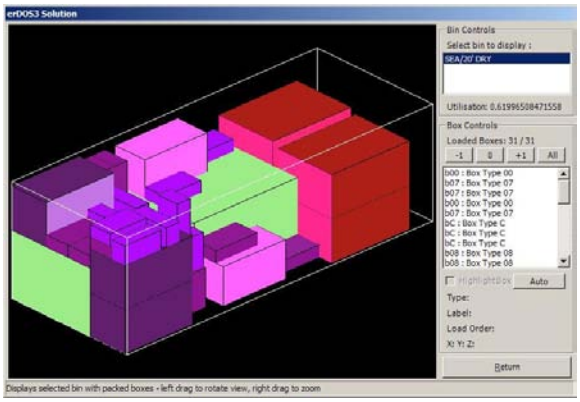


Figure 20: *Complex problems give questionable results*

In all, the software could benefit from some slight improvement – perhaps redesigning the form layouts to improve readability, and implementing keyboard shortcuts for view controls. Another possible area that could benefit from some attention is that of the box appearance. The software uses solid colours, generated at random during the box creation, for shading and representing the box in the 3-dimensional view. By implementing some textures into the boxes, the final display could be improved further.

References

[Ast06] AstroKettle.: http://www.astrokettle.com

[JCH00] Jiang Y. D., Cha J. Z., He D. Y.: Research on loading rectangular freight into a container. *J. of the China Railway Society* (2000), vol. 22, no. 6, pp. 13-18.

[Mag06] MagicLogic.: http://www.magiclogic.com

[Ope06] OpenGL.: http://www.opengl.org

[Pis02] Pisinger D.: Heuristics for the container loading problem, *European Journal of Operational Research*, (2002), vol. 141, no. 2, pp. 382-392.

[GR81] George J. A., Robinson D. F.: A heuristic for packing boxes into a container, *Computers and Operations Research* (1980), vol. 7, pp. 147-156.

[TAO06] TAO .NET Framework.: http://www.mono-project.com/tao