Appears in Proc. Intl. Workshop on Planning & Scheduling for Space, Moffett Field, CA, March 2013.

Automating Stowage Operations for the International Space Station

Russell Knight, Gregg Rabideau, Andrew Mishkin, Young Lee

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Russell.Knight@jpl.nasa.gov
Gregg.Rabideau@jpl.nasa.gov
Andrew.Mishkin@jpl.nasa.gov
Young.H.Lee@jpl.nasa.gov

Abstract

Stowage operations (the act of storing and retrieving items) onboard the International Space Station takes up approximately 25% of each astronaut's time. Until recently, managing stowage was fundamentally performed by hand by extremely skilled technicians called stowage officers. These individuals need to be able to know, by simply examining a list of items and a location, how much room is left at a location and if some new item could be placed there. We have provided a fielded capability that uses a novel box-packing algorithm combined with a database search capability to aid stowage officers in their duties.

Introduction

This paper is organized as such:

1. brief ISS stowage description
2. fast item lookup
3. box packing
4. results
5. related work

ISS Stowage Description

Managing stowage is fundamentally managing the inventory flying onboard the International Space Station. A database detailing the contents of each location is used and updated when the contents change location, are used up, are lost, or occasionally when they are discovered. As part of the Automating and Streamlining ISS Mission Operations (ASIMO) collaboration between the Jet Propulsion Laboratory and Johnson Space Center, our team examined improving the stowage processes. Through the use of Certification Based Analysis [Knight 2010], we determined a set of processes that could be improved if we could reduce the number of times the products failed certification (i.e., required further work or failed to be what the user intended). Three of these processes would benefit from a capability to automatically find locations in which to store items. This paper focuses on the technology delivered for these processes.

The three processes we focus on are 1) location suggestion for found material, 2) Prepack list generation, and 3) Unpack list generation.

Location Suggestion for Found Material

Location suggestion for found material is a two-step process. This scenario is that an item has been identified onboard station and a location needs to be found that accommodates storing the item. The ground personnel must work with the astronaut to 1) identify the item and then 2) determine an appropriate location.

The identification of the item using the database management system is usually straight-forward, but we saw the opportunity to improve on this by providing an interface that allowed any sort of information to be entered and all fields would be searched automatically on every keystroke. For example, if an astronaut finds a object, we could use a description fragment, e.g., “ctb divider” to get a list of candidate parts that the found item could be. We could then add details, such as a part-number fragment e.g., “ctb divider seg3311” which would result in the list being narrowed down. Of course, we could use the barcode, but this is not always on the item. Details on our approach can be found in the Fast Item Lookup section of this paper.

This work was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Once the item is identified, we have an exemplar to search for locations. We could simply place the item in the location that it is supposed to be in, or we might need to find a new home for it. Once we have identifying information, such as part number and who owns it, as well as dimensional information. Since stowage officers prefer storing similar items together, this is very important information. We then make use of our Box Packing technology to suggest several locations to place the item.

Prepack List Generation

Prepack list generation is the process of making a list of items to be shipped out of the ISS. This list includes bags to be used and what items are to go into which bags. This can be quite lengthy, and needs to be updated as the situation changes. To achieve automation of the prepack list generation, we use our automated box packing algorithm to pack the bags with the items that are to be shipped out. But, we also want to accommodate manual selection of items and addition of items to the list. This is aided by our fast item lookup algorithm.

Unpack List Generation

Unpack list generation is the process of making updating a list of items that are to be shipped to the ISS with the appropriate locations onboard. This list not only included items, but also information, such as crew preference information. Crew preference items are those that should be stowed in the appropriate crew preference locations; clearly we want the right underwear to go to each crew member. As with prepack list generation, we automate unpack list generation by using our box-packing algorithm to find locations onboard station for all items being stowed. Of course, manual selection of destinations is aided by our fast item lookup capability.

Fast Item Lookup

Our approach to fast item lookup was to pre-compile search substrings into a lookup table, and the perform joins across the results using set intersection in memory. This greatly outperforms database query (which drastically slowed typing of queries). Typing of any entry results of instantaneous real-time list updates. Note that this is a modest database of approximately one hundred thousand entries and we search across ten fields.

As a search entry is entered, a set of keys is generated by breaking up the entry at white space borders. Each key is then considered to be a filter, and a set of data entries is generated that has at least one field that starts with the key. Since this is incremental, the worst case is the generation of the list for the very first key. As the key is extended, the list is pared down. When a new key is introduced, a new list is generated by first copying the set of entries from the previous key. This allows users to backspace over entries without having to recompile the lists, although backspacing through a partial entry does require repopulation, thus we only perform this when a user has let an updated (partially deleted) key idle for more than 1/2 of a second. Because of how we cascade key filter lists, the resultant list of items that represents the intersection of all lists is simply the filter list of the last key.

Box Packing

We find it surprising that, even though it seems an obvious problem to address, very little work is available in the literature regarding packing a set of 3-dimensional items into a single 3-dimensional container where the dimensions of the container are fixed. Most problem characterizations, such as rectangle packing problems, focus on selecting the container of smallest dimension that accommodates the entire list of items. Other approaches fail for small numbers of boxes (20) or provide loose guarantees on quality [Miyazawa and Wakabayashi].

We often have many items in a container (over 30) that we need to search for and find a set of solutions within a very short period of time (preferably less than 10 seconds). Also, the raw branching-factor for 3-dimensional box packing is very high. Each box can be in one of 6 orientations, and every placed box removes one candidate location, but introduces 3 more, on average. This results in a branching factor of at least 12, but can be worse if more positions are induced due to adjacencies of other previously placed boxes. All of these factors lead us to believe that optimal techniques might not lead us to the best solutions for our problem.

To reiterate, we have a single 3-dimensional rectangular solid (a box) that we wish to find a home for. We look through all of the candidate locations onboard station (also characterized as 3-dimensional rectangular solids). To determine whether or not a box can fit in a container, we add the box to the list of contents for that container, and then try to pack the entire list into the box.

Our solver works as such:

1. Order the items in descending order of the sum of the squares of the length, width, and height. Assign priorities as such.
2. For the first 5 boxes, search exhaustively for a solution. If a solution is found, carry on to the next step, but allow for backtracking if any of the subsequent steps fail
3. In priority order, attempt to fit each item in the first location that it will fit in the container. The orientation chosen should be in order of the

orientation that fits the most items of that size in the container.

4. If any items fail to be placed in the container, increase their priority by *n*, where n is the number of items.
5. If we have attempted to reorder the items and failed to completely pack the container 100 times, fail to line 2.

Ordering the items in descending order of the sum of the squares of the length, width, and height was found to be greatly superior to simply ordering by volume or largest dimension or sum of dimensions. We hypothesize that this is due to faithfully identifying hard things pack, like long poles. This also scales with the interior diagonal, which we think faithfully represents the amount of inflexibility that is introduced by placing the item in the box.

The short exhaustive search turned out to be very helpful in that there often were a few large items in locations that needed to be carefully placed, and many smaller items that could fit just about anywhere. The value of 5 might seem somewhat magical, but consider that there are over one hundred thousand positions to consider with 5 boxes being searched exhaustively (approximately 124,416), but over one million positions to be considered 6 boxes (approximately 1,492,992). Empirically, the reduced performance for one more box wasn't justified.

The iterative priority-based optimization allowed us to quickly identify problem boxes and promote their placement earlier in the packing sequence. Again, this is due to having some problematic items accompanying many smaller trivial items. Also, keeping the priority information between runs where we changed the orientation and placement of some of the first five items allowed us to continue progressing through the space of item orderings.

Finally, choosing orientations first that optimize packing as if that were the only item and we were trying to fill the container with the item was helpful in that we often had a great deal of smaller items that were to be packed into a location. This heuristic ensures that a good deal of these could be packed in the case that we were simply packing 200 filters someplace.

A short note about bags: when packing a cargo transfer bag, the bag is rather amorphous until it starts getting full, then it starts resembling a rectangular solid, thus we use the rectangular solid dimensions to characterize large bags. Small bags, such as Ziploc bags, often remain amorphous. To correctly characterize packing these, the items are removed from the bags and placed in the list individually to be packed. We rely on the astronauts to figure out how to squeeze the contents around in practice.

Results

Our results are purely in the form of the delivered system and our empirical observations on its performance. This system provides the required functionality for Location Suggestion, Prepack list generation, and Unpack list generation.

Figure 1 shows the entry window and partial search results for our fast lookup function for location suggestion. To select any of these, we press the select button, resulting in the view in Figure 2. Here we select to find a location for up to 10 items of the same type (in this case, a cargo transfer bag divider). We kick of the search by selecting the *Get next locations* button. We are then presented with a list of candidate locations, with the number of items accommodated by the location in parenthesis (Figure 3). If we select one of these, we can view the various packings, as shown in Figure 4 (left view), Figure 5 (right view), and Figure 6 (top view). In each of these views, the bright yellow items are the dividers that we are trying to place.

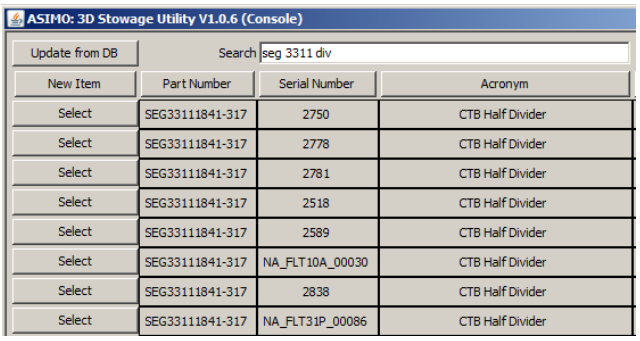


Figure 1 Location Suggestion: Item Lookup

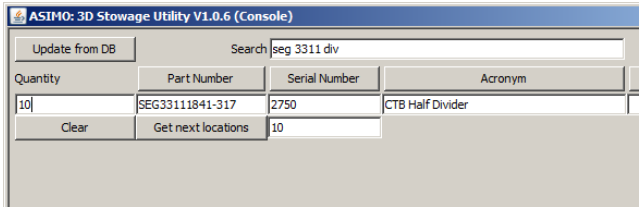


Figure 2 Location Suggestion: setting quantity

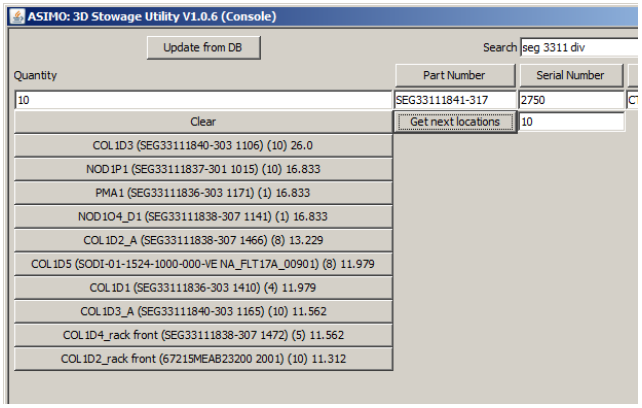


Figure 3 Location Suggestion: search results

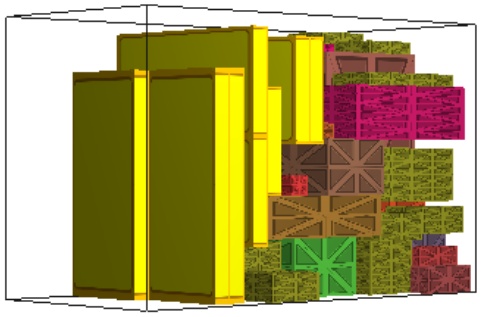


Figure 4 Location Suggestion: Left view of one candidate location

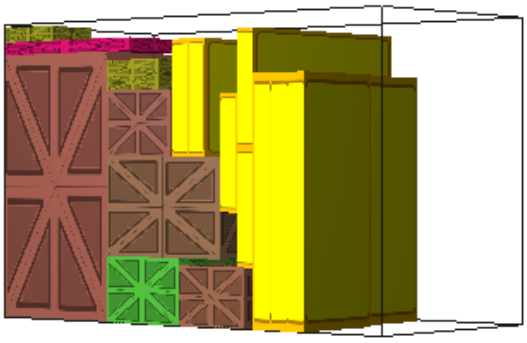


Figure 5 Location Suggestion: Right view of one candidate location

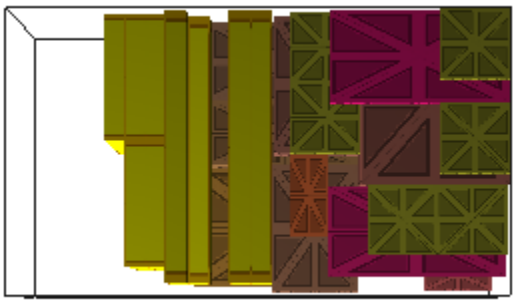


Figure 6 Location Suggestion: top view of one candidate location

Our prepack implementation allows for ingesting a Microsoft Excel file and adding bags to be prepacked. Figure 7 shows the display of such a list. Note that items in yellow and red indicate problems with the input, such as serial numbers not yet being assigned. Nonetheless, we can pack these items automatically in a matter of seconds into the bags, as shown by the Figure 8.

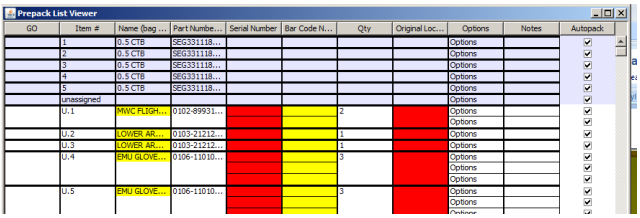


Figure 7 Prepack List: before packing

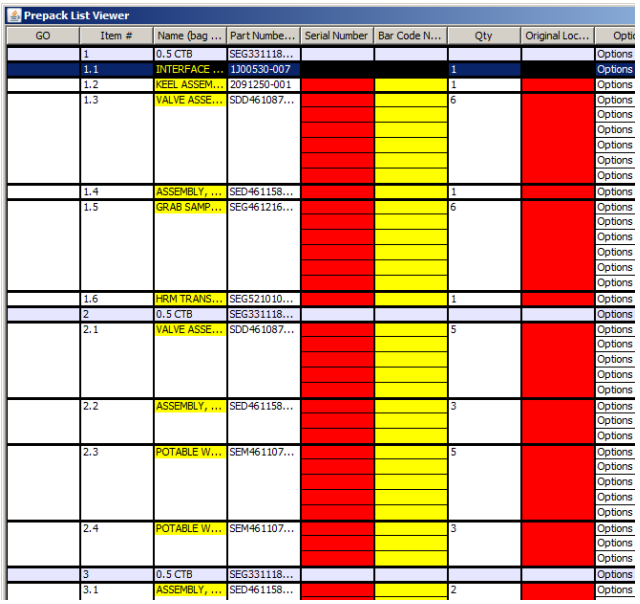
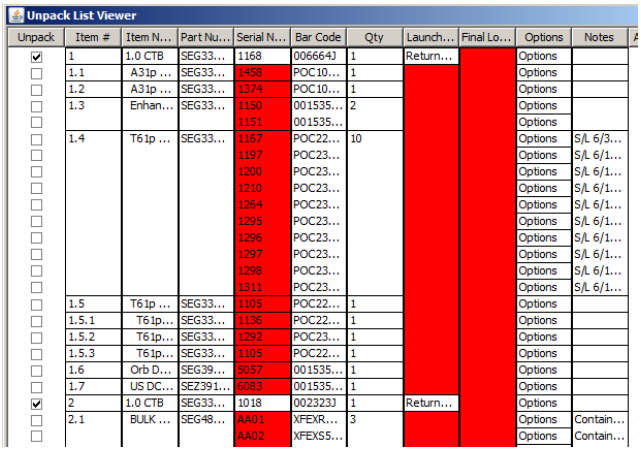


Figure 8 Prepack List: after packing

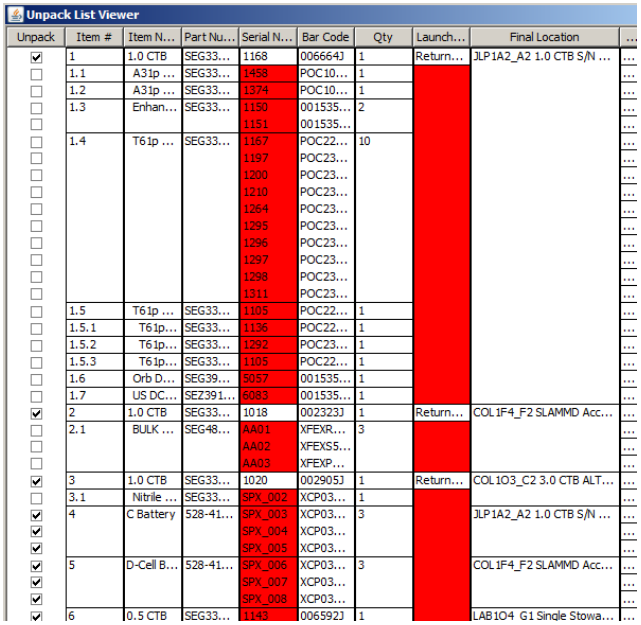
Similarly, our unpack implementation allows for ingesting a Microsoft Excel file and assigning locations to the items that are in the list. Figure 9 shows such a list. Items in red indicate that the field for the item in the list does not match any entry in the database. Nonetheless, we can still automatically pack these into locations onboard the ISS in a matter of seconds, as shown by the assigned Final Locations in Figure 10.



Unpack List Viewer

Unpack	Item #	Item N…	Part Nu…	Serial N…	Bar Code	Qty	Launch…	Final Lo…	Options	Notes
☑	1	1.0 CTB	SEG33…	1168	006664J	1	Return…		Options	
☐	1.1	A31p …	SEG33…	1458	POC10…	1			Options	
☐	1.2	A31p …	SEG33…	1374	POC10…	1			Options	
☐	1.3	Enhan…	SEG33…	1150	001535…	2			Options	
☐				1151	001535…				Options	
☐	1.4	T61p …	SEG33…	1167	POC22…	10			Options	S/L 6/3…
☐				1197	POC23…				Options	S/L 6/1…
☐				1200	POC23…				Options	S/L 6/1…
☐				1210	POC23…				Options	S/L 6/1…
☐				1264	POC23…				Options	S/L 6/1…
☐				1295	POC23…				Options	S/L 6/1…
☐				1296	POC23…				Options	S/L 6/1…
☐				1297	POC23…				Options	S/L 6/1…
☐				1298	POC23…				Options	S/L 6/1…
☐				1311	POC23…				Options	S/L 6/1…
☐	1.5	T61p …	SEG33…	1105	POC22…	1			Options	
☐	1.5.1	T61p…	SEG33…	1136	POC22…	1			Options	
☐	1.5.2	T61p…	SEG33…	1292	POC23…	1			Options	
☐	1.5.3	T61p…	SEG33…	1105	POC22…	1			Options	
☐	1.6	Orb D…	SEG39…	5057	001535…	1			Options	
☐	1.7	US DC…	SEZ391…	6083	001535…	1			Options	
☑	2	1.0 CTB	SEG33…	1018	002323J	1	Return…		Options	
☐	2.1	BULK …	SEG48…	AA01	XFEXR…	3			Options	Contain…
☐				AA02	XFEXS5…				Options	Contain…

Figure 9 Unpack List: before packing



Unpack List Viewer

Unpack	Item #	Item N…	Part Nu…	Serial N…	Bar Code	Qty	Launch…	Final Location	…
☑	1	1.0 CTB	SEG33…	1168	006664J	1	Return…	JLP1A2_A2 1.0 CTB S/N …	…
☐	1.1	A31p …	SEG33…	1458	POC10…	1			…
☐	1.2	A31p …	SEG33…	1374	POC10…	1			…
☐	1.3	Enhan…	SEG33…	1150	001535…	2			…
☐				1151	001535…				…
☐	1.4	T61p …	SEG33…	1167	POC22…	10			…
☐				1197	POC23…				…
☐				1200	POC23…				…
☐				1210	POC23…				…
☐				1264	POC23…				…
☐				1295	POC23…				…
☐				1296	POC23…				…
☐				1297	POC23…				…
☐				1298	POC23…				…
☐				1311	POC23…				…
☐	1.5	T61p …	SEG33…	1105	POC22…	1			…
☐	1.5.1	T61p…	SEG33…	1136	POC22…	1			…
☐	1.5.2	T61p…	SEG33…	1292	POC23…	1			…
☐	1.5.3	T61p…	SEG33…	1105	POC22…	1			…
☐	1.6	Orb D…	SEG39…	5057	001535…	1			…
☐	1.7	US DC…	SEZ391…	6083	001535…	1			…
☑	2	1.0 CTB	SEG33…	1018	002323J	1	Return…	COL1F4_F2 SLAMMD Acc…	…
☐	2.1	BULK …	SEG48…	AA01	XFEXR…	3			…
☐				AA02	XFEXS5…				…
☐				AA03	XFEXP…				…
☑	3	1.0 CTB	SEG33…	1020	002905J	1	Return…	COL1O3_C2 3.0 CTB ALT…	…
☐	3.1	Nitrile …	SEG33…	SPX_002	XCP03…	1			…
☑	4	C Battery	528-41…	SPX_003	XCP03…	3		JLP1A2_A2 1.0 CTB S/N …	…
☑				SPX_004	XCP03…				…
☑				SPX_005	XCP03…				…
☑	5	D-Cell B…	528-41…	SPX_006	XCP03…	3		COL1F4_F2 SLAMMD Acc…	…
☑				SPX_007	XCP03…				…
☑				SPX_008	XCP03…				…
☑	6	0.5 CTB	SEG33…	1143	006592J	1		LAB1O4 G1 Single Stowa…	…

Figure 10 Unpack List: after packing

The real measurement of performance for this system is in the reduction of errors on console and the reduction in errors and workforce for the production of prepack and unpack lists. Although this application has been online for only a short period of time, stowage officers have reported significant reductions in on-console errors having to do with suggesting locations to astronauts for stowage. Similarly, the number of people required for prepack and unpack list generation has been halved, but it isn't clear yet what the actual impact is due to the retirement of the space shuttle. Now that SpaceX is delivering payloads to the ISS, we will soon be able to report on performance improvements under equivalent or increased workloads.

Related Work

With respect to rectangle packing, Huang and Korf provide an optimal system in the 2-dimensional space, but again this is with respect to an adjustable container. Conversations with Eric Huang led us to believe that while these techniques might work for smaller instances, the increased branching factor in a 3-d space would need to be addressed.

Miyazawa and Wakabayashi provide an algorithm with guarantees on worst case performance of a factor of 2.67, but in practice our approach performs better. This is likely due to slack introduced by the forced iterative leveling applied to layers of boxes, the stacking of like-sized items in columns thereby precluding their use as fillers, and possibly due to the problematic cases of bad orientations for large collections of homogenous boxes. Future work should compare these approaches side by side.

Clement et al provided a prototype system that addressed spatial constraints with respect to the storage of items onboard the ISS, but this had to do with planning operations and moving items to ensure pathways, not with the actual task of stowage. Very little was implemented that optimized the packing of the space.

Acknowledgements

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology under contract with NASA.

We acknowledge the tireless efforts of the ISS Stowage team at the Johnson Space Center. Their deep knowledge and practical advice aided greatly in the delivery of this system. Specifically we acknowledge the efforts of Ursula Stockdale, Casey Johnson, Robert Adams, Kary “Scott” Smith, Larry “Joey” Crawford, Margaret Gibb, and Roger Galpin.

References

Bradley J. Clement, Michael J. Iatauro, Javier Barreiro, Russell Knight, and Jeremy D. Frank. "Spatial Planning for International Space Station Crew Operations." In Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-10). Sapporo, Japan, August, 2010.

Eric Huang, Richard E. Korf. "Optimal Rectangle Packing: An Absolute Placement Approach." J. Artif. Intell. Res. (JAIR) 46: 47-87 (2013)

Russell Knight. "Technology Infusion via Certification-based Analysis." In Proceedings of the 10th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-10). Sapporo, Japan, August, 2010.

F. K. Miyazawa and Y. Wakabayashi, “An algorithm for the three-dimensional packing problem with asymptotic performance analysis,” Algorithmica, May 1997, volume 18, Issue 1, pp 122-144.