Automatic Exploded View

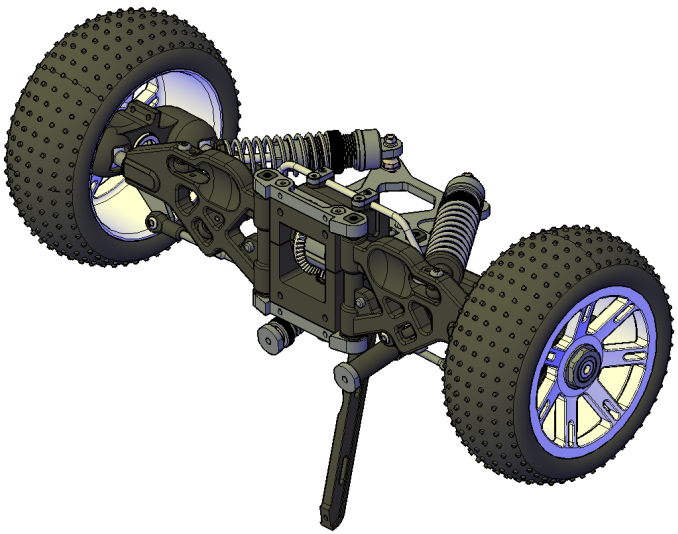
Kirill Brodt

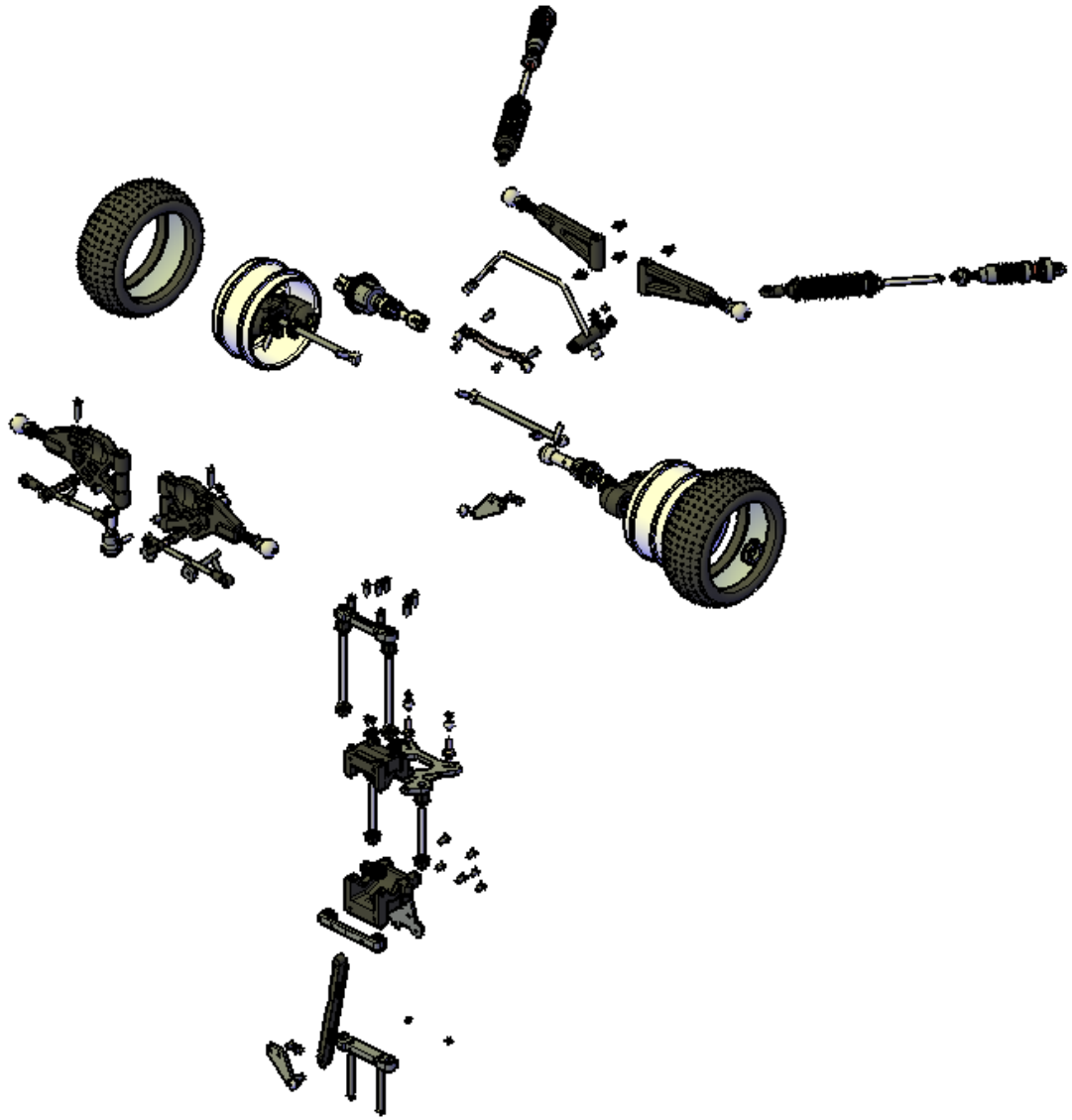
Bricsys

May, 2018

Motivation

BRICSCAD





NP complete…

Outline

Section 1

Work accomplished

Work accomplished

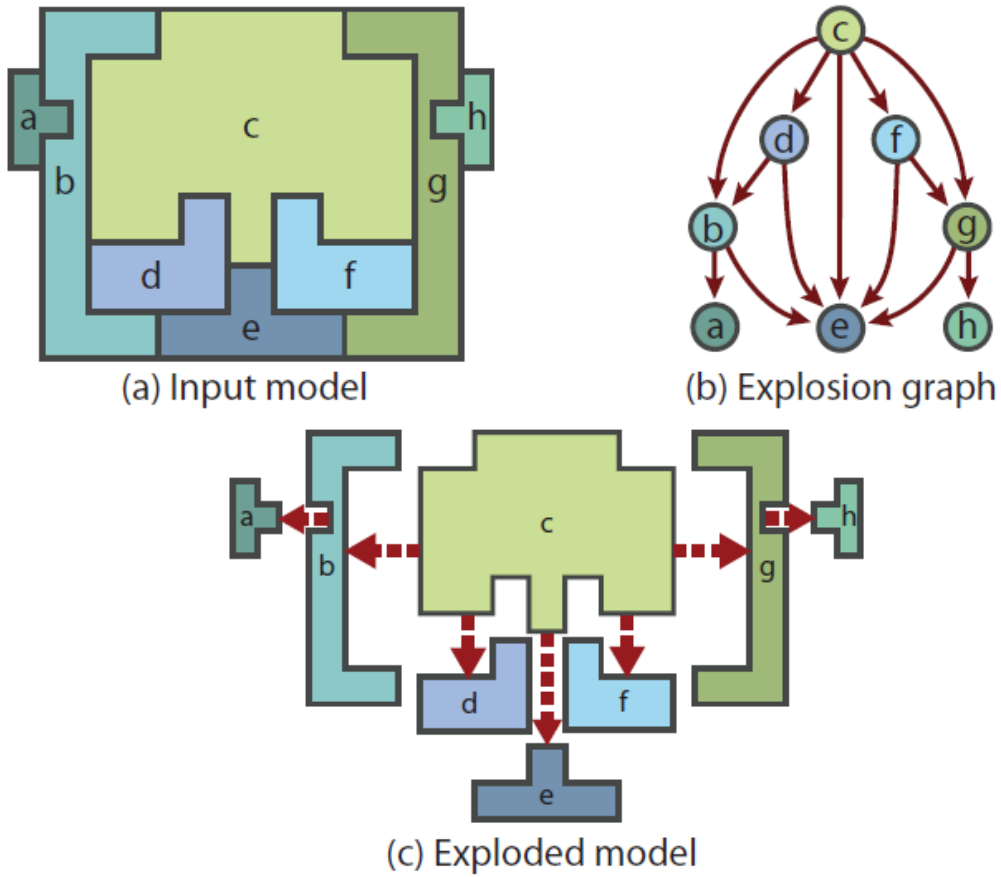
- Research
  - Literature review and compilation
  - Algorithm of exploded view
- Technologies
  - C++
  - Visual Studio 2013
  - ACIS (geometric modeling kernel)
- Architecture
  - `class Part;` (`getBoundingBox`, `transform`, etc...)
  - `class ExplodedViewAlgorithm;` (abstract class)
  - `class LinearExplodedViewAlgorithm;`
  - `class AutomaticExplodedViewAlgorithm;`
  - `class HierarchicalExplodedViewAlgorithm;`

Section 2

Literature review

Literature review

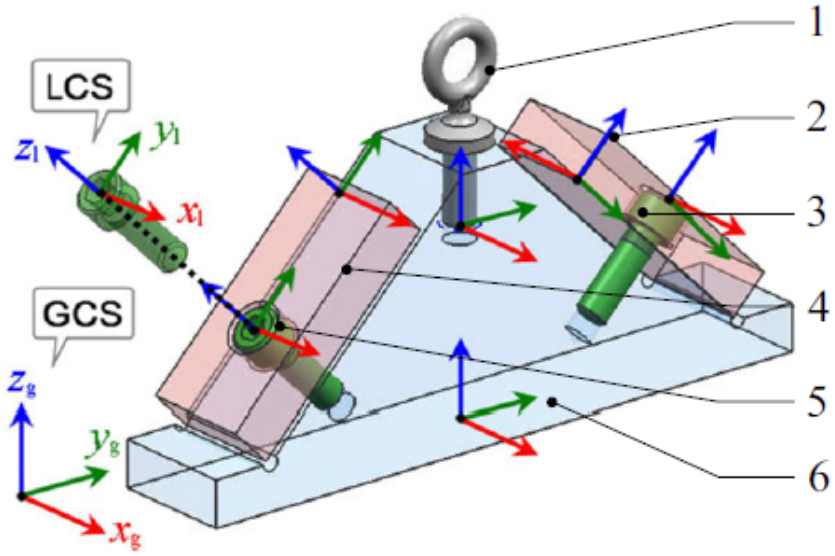
In[1] authors propose to construct a directed acyclic explosion graph representing the relative order in which parts can be exploded without violating blocking constraint (possible explosion directions obtained by local translational freedom (LTF) cones).



[1] Li, Agrawala, Curless, *et al.*, "Automated Generation of Interactive 3D Exploded View Diagrams", 2008

Literature review

In[2] authors propose to use matrix-based assembly models representing the relationships between parts for 12 directions (6 axes of global and 6 axes of local coordinate systems).



[2] Yu and Zhang, “Hierarchical exploded view generation based on recursive assembly sequence planning”, 2017

Basic approach

```
struct Part
{
  BBox getBoundingBox(coordinate_system);
  void move(direction);
  bool isBlockedBy(target_part, direction);

  // some other stuff...
};
```

Basic approach

Algorithm 1 Disassembly sequence construction

Require: Set $\mathtt{S}$ of active parts (unremoved parts)
Ensure: Disassembly sequence set $\mathtt{D}$

1: $\mathtt{D} \leftarrow \emptyset$
2: **while** $\mathtt{S} \neq \emptyset$ **do**
3: $\quad \mathtt{C} \subset \mathtt{S} \leftarrow \mathtt{getCandidateSet}(\mathtt{S})$ ▷ unblocked parts in at least one direction
4: $\quad \mathtt{P}^* \subset \mathtt{C} \leftarrow \mathtt{chooseBestParts}(\mathtt{C})$ ▷ choose «best» parts to remove
5: $\quad \mathtt{D} \leftarrow \mathtt{D} \cup \mathtt{P}^*$ ▷ add $\mathtt{P}^*$ to disassembly sequence set
6: $\quad \mathtt{S} \leftarrow \mathtt{S} \setminus \mathtt{P}^*$ ▷ remove part
7: **end while**

- `getCandidateSet` uses `isBlockedBy`
- `chooseBestParts` depends on specification

Basic approach

Algorithm 2 Exploded view

Require: Disassembly sequence set `D`
Ensure: Exploded view
1: accumulated bounding box ← `D.first.getBoundingBox()`
2: **for** part ∈ `D` **do**
3: current bounding box ← part.`getBoundingBox()`
4: move part to escape accumulated bounding box
5: enlarge accumulated bounding box by current bounding box
6: **end for**

Section 3

Linear Exploded View

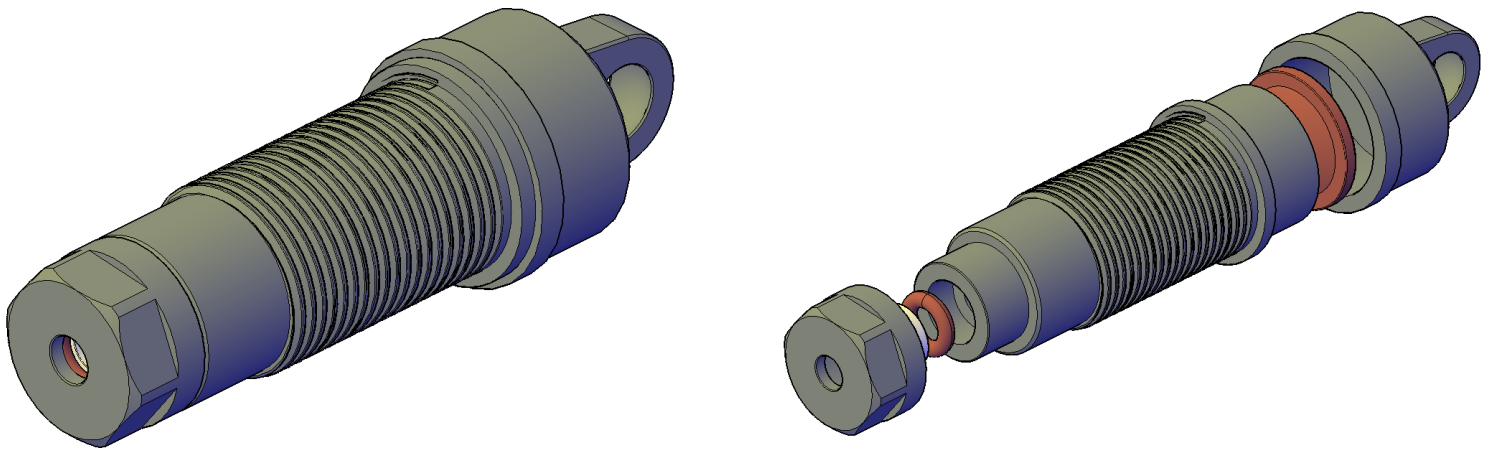
Linear Exploded View

- Given direction d define the matrix E also called extended interference matrix (EIM):

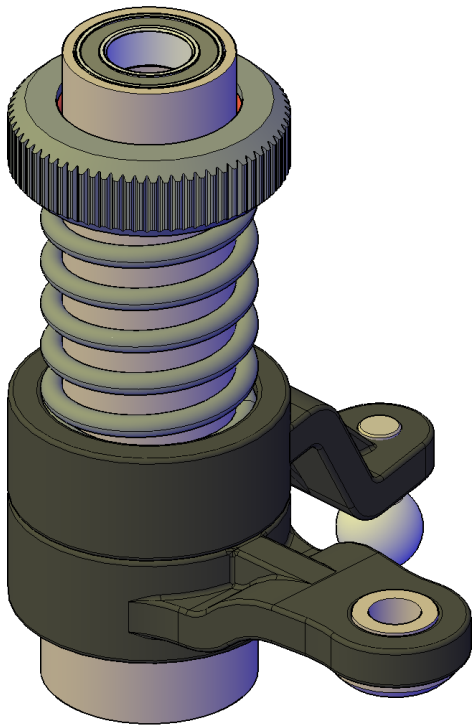
$$e_{ij} = \begin{cases} 0, & \text{if } p_i \text{ does not interfere with } p_j, \\ 1, & \text{otherwise.} \end{cases} \leftarrow \texttt{isBlockedBy}$$

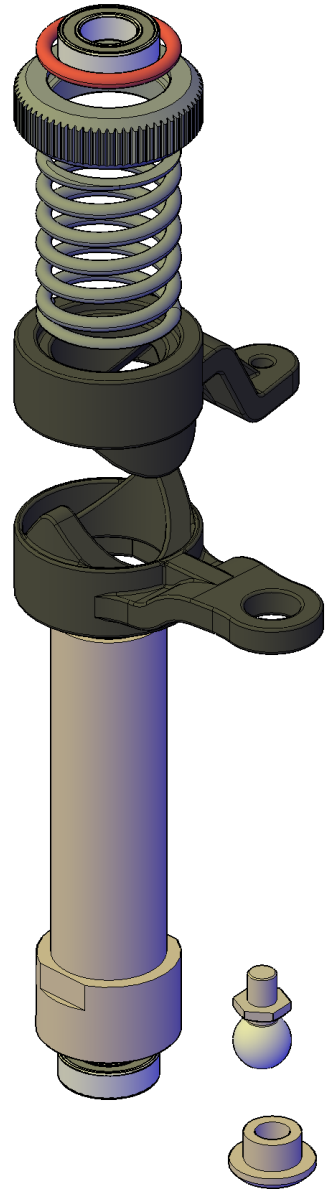
- If $E_{i\bullet} = 0$ then part p_i is removable $\leftarrow$`getCandidateSet`
- If there is no such parts let's consider the matrix E as a directed graph and find all strongly connected components to condensate into the tree.
- Consider all parts from one connected component as a whole part $\leftarrow$`chooseBestPart` and add them to the disassembly sequence set.
- Explosion follows the rule of "the later disassembled components explode earlier" $\leftarrow$`getBoundingBox, move`

Results of Linear Exploded View

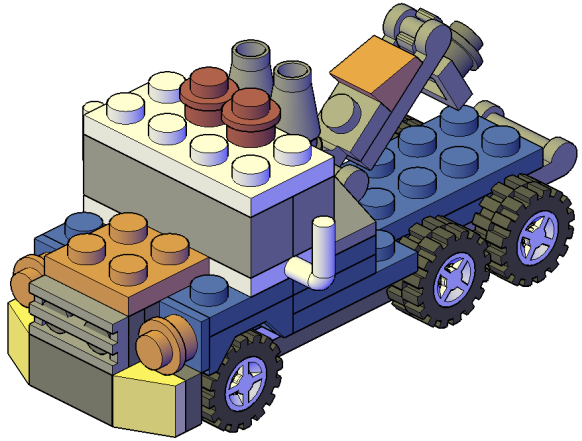


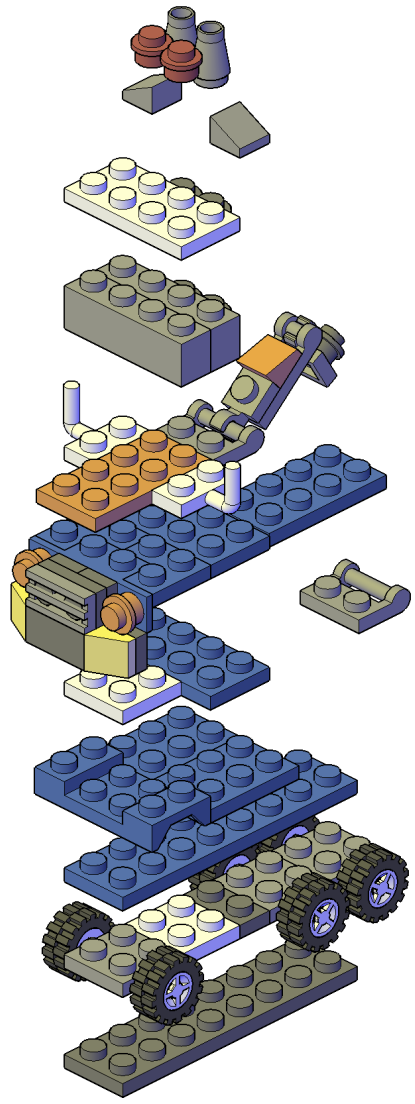
Results of Linear Exploded View



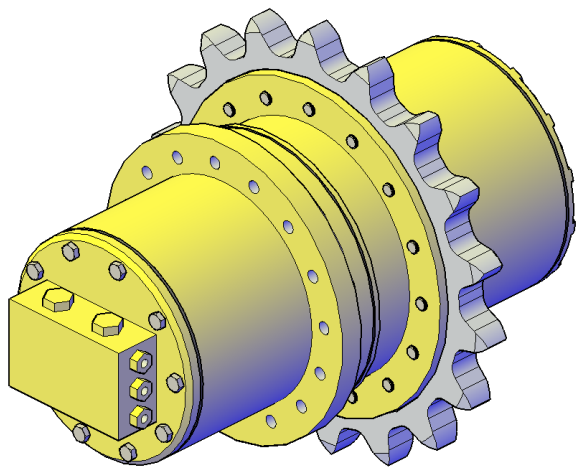


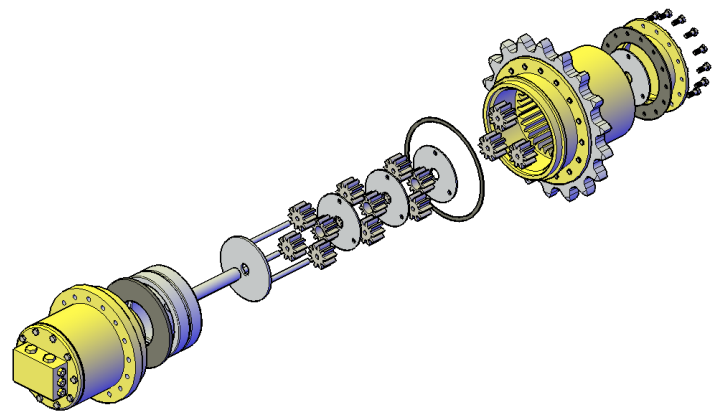
Results of Linear Exploded View





Results of Linear Exploded View





Drawbacks of Linear Exploded View

- To construct the matrix E we need $O(n^2 b)$ time complexity with $n =$ number of parts and $b =$ complexity of `isBlockedBy` depending on geometry.
- `isBlockedBy` uses ray tracing emitting from all vertices of surface finite element mesh of the part to test the interference.
- So this function sometimes gives false positives and false negatives. Thus it must be improved both in performance and accuracy.

Assembly	Number of parts	CPU time in seconds
Leng	7	2.5
Szerv	10	16
Loader	61	7.5
Gem	73	54

- i7-7700HQ CPU @ 2.80GHz
- Approximately time complexity is $O(n^3 mc + n^2 b)$, $m =$ number of edges in graph and $c =$ number of independent sets.

Section 4

Automatic Exploded View

Automatic Exploded View

- Let's generalize previous linear exploded view algorithm for general case.
- Take six axial directions of the global coordinate-system $\pm x, \pm y, \pm z$ but, we go further and take also six axial directions of the local coordinate systems of the components. Construct EIM matrices for each direction and use the same algorithm as in linear case.

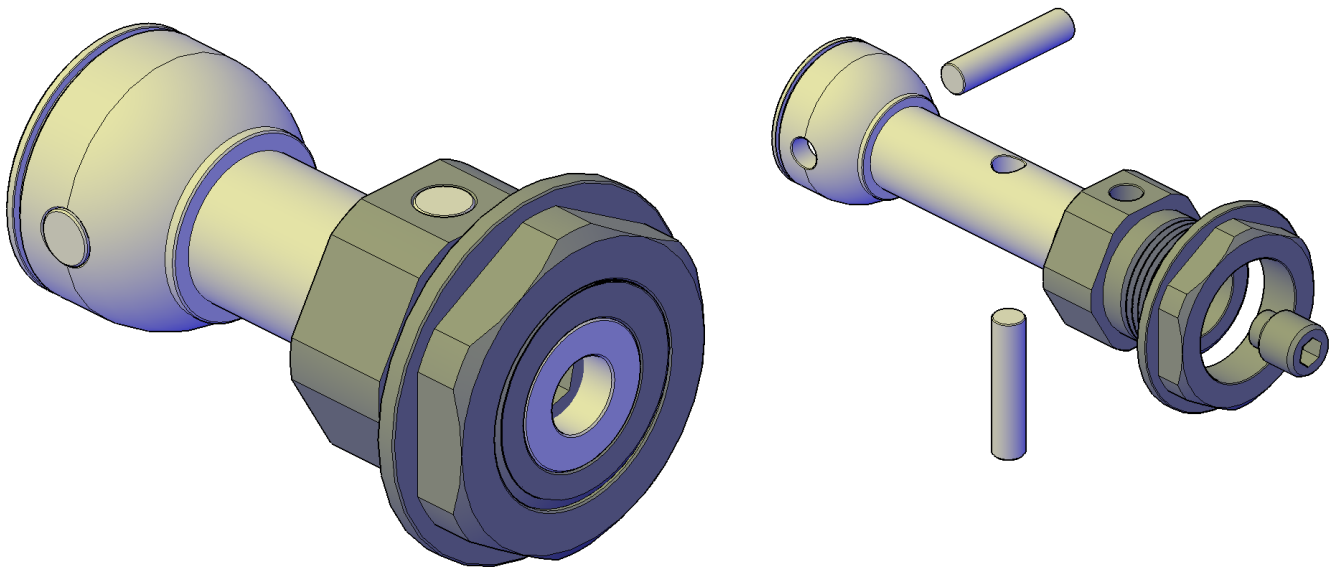
But how to choose the "best" parts to remove from candidate set?

Simply take the direction with the most number of unblocking parts to move them simultaneously.

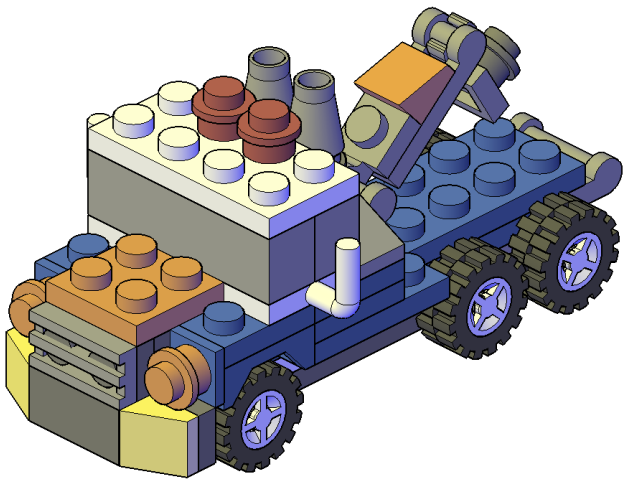
- PROFIT!

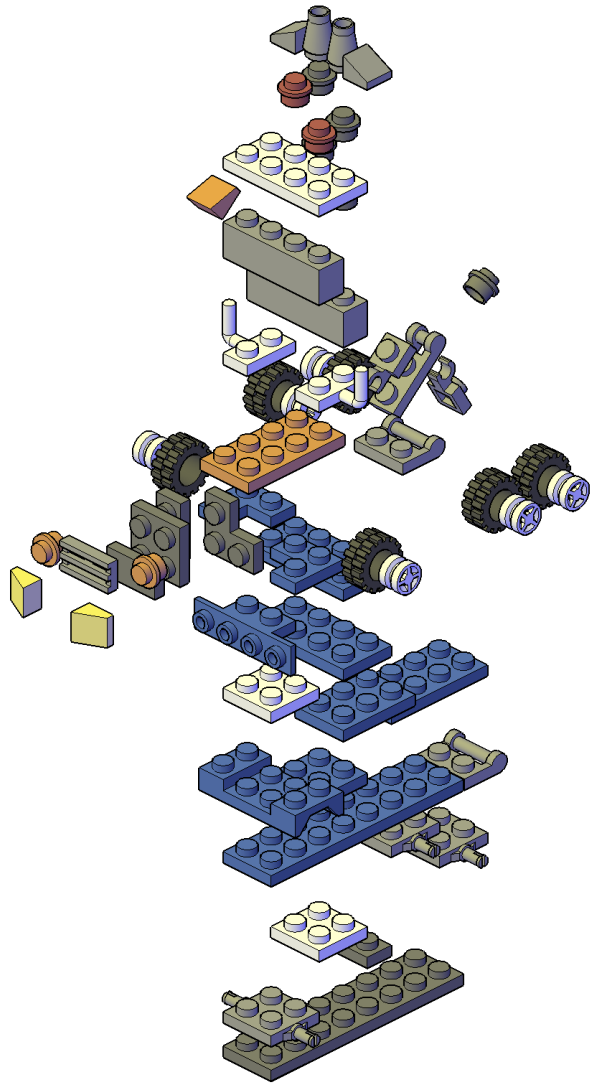
Unblocking parts such that theirs bounding boxes do not overlap $\Rightarrow$ maximum disjoint set (MDS), another NP complete problem inside NP complete problem :)

Results of Automatic Exploded View



Results of Automatic Exploded View





Drawbacks of Automatic Exploded View

- The same as for Linear Exploded View.
- From the set of all possible directions choose the most natural direction.
- Very slow, need to be constructed 12(!) EIM matrices
Assembly	Number of parts	CPU time in seconds
Itengely	6	3.2
Loader	61	27
- Approximately time complexity is $O(12 \times n^3 mc + 9 \times n^2 b)$,

n = number of parts, m = number of edges in graph,
c = number of independent sets and b = complexity of `isBlockedBy`.

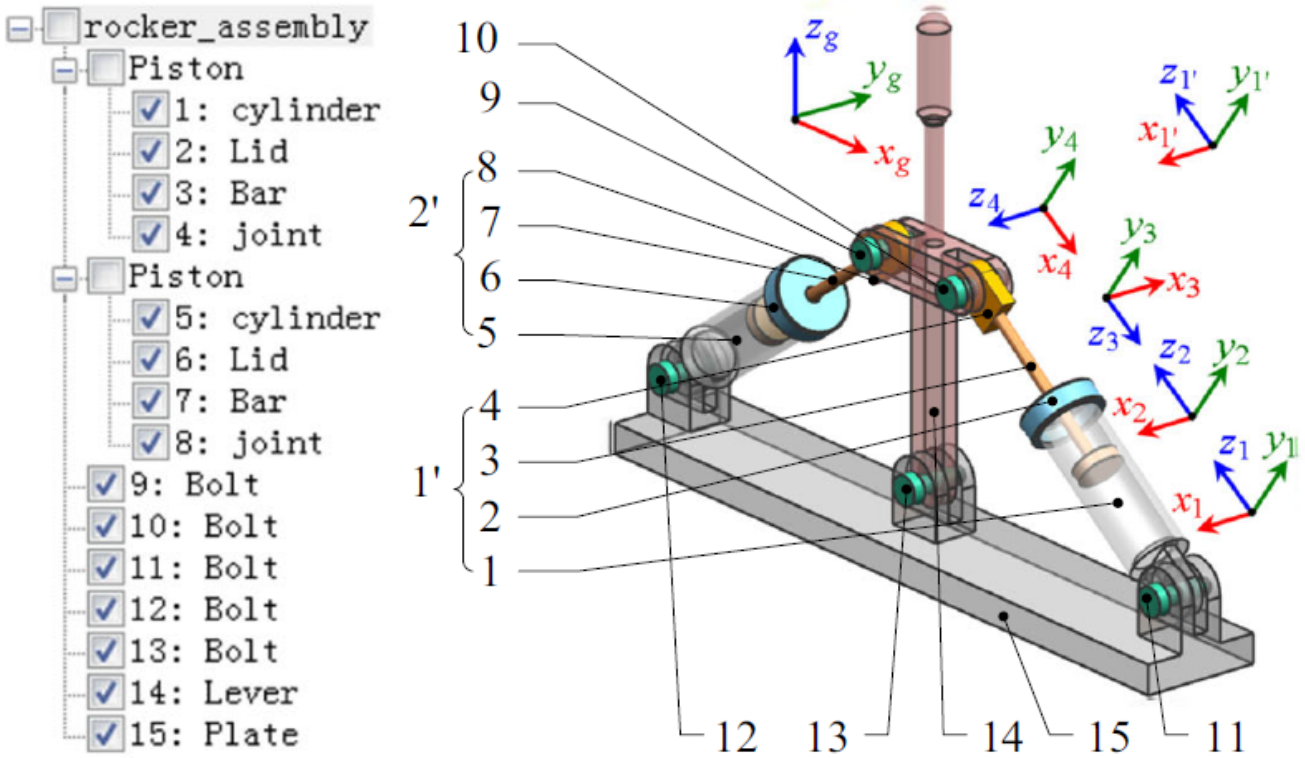
Section 5

Hierarchical Exploded View

Hierarchical Exploded View

Idea!

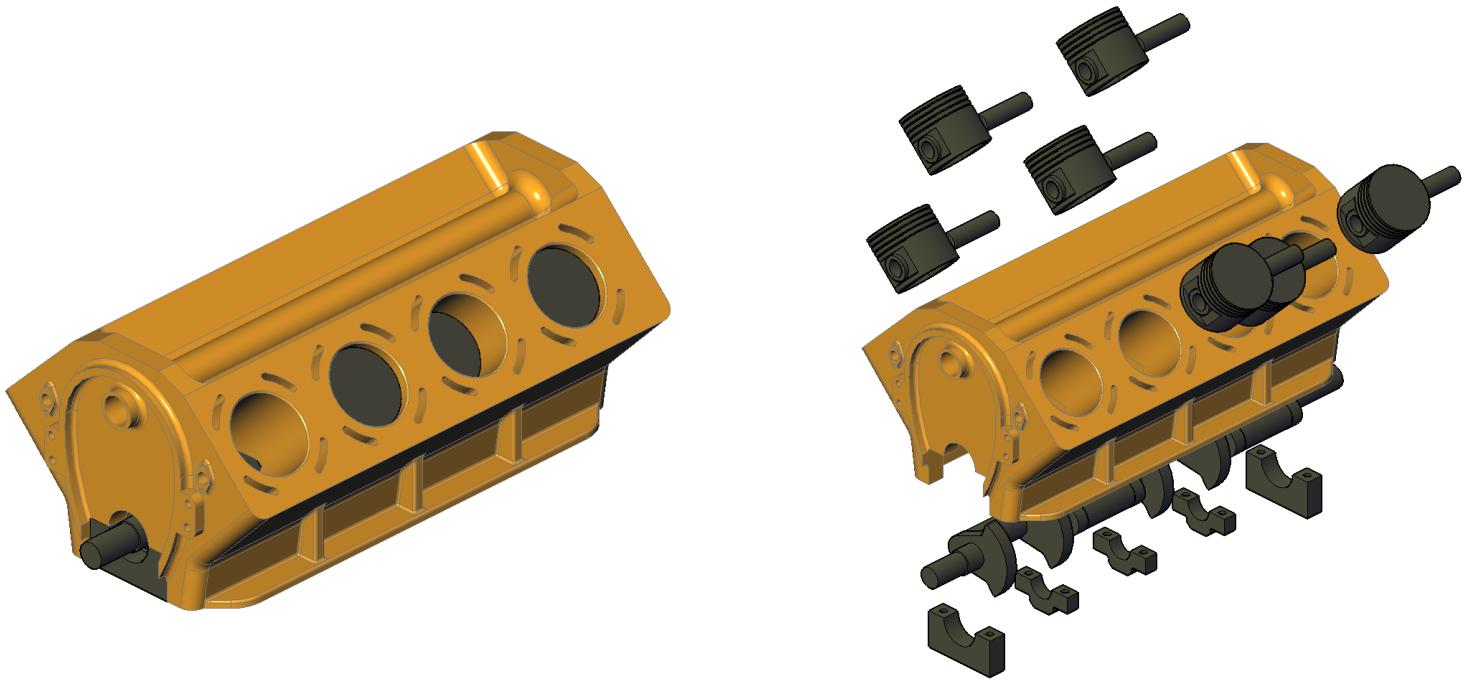
What if we use hierarchical exploded view relying on assembly tree?



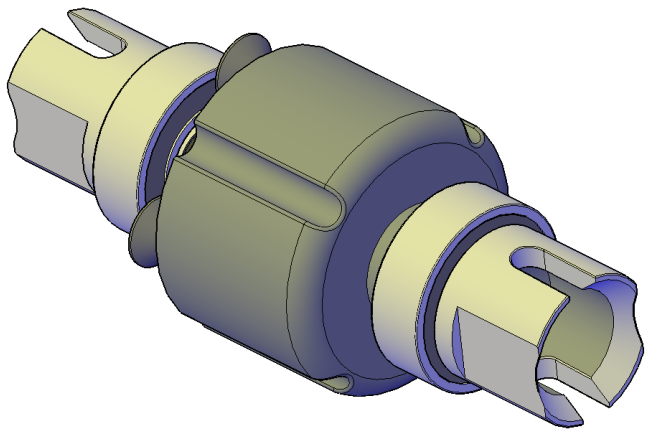
Hierarchical Exploded View

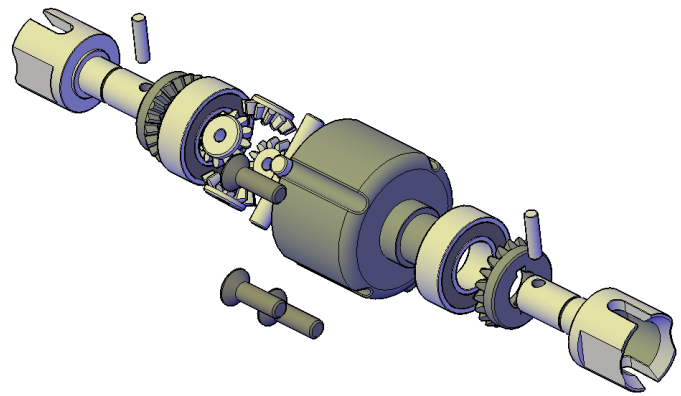
- Using assembly tree, we can start from the bottom levels representing sub-assemblies of full assembly and do explosion view with previous algorithm
- Raising to higher levels only the base part is taken instead of whole sub-assembly from lower levels
- Bounding box of this base part is calculated as the bounding box of whole sub-assembly

Results of Hierarchical Exploded View

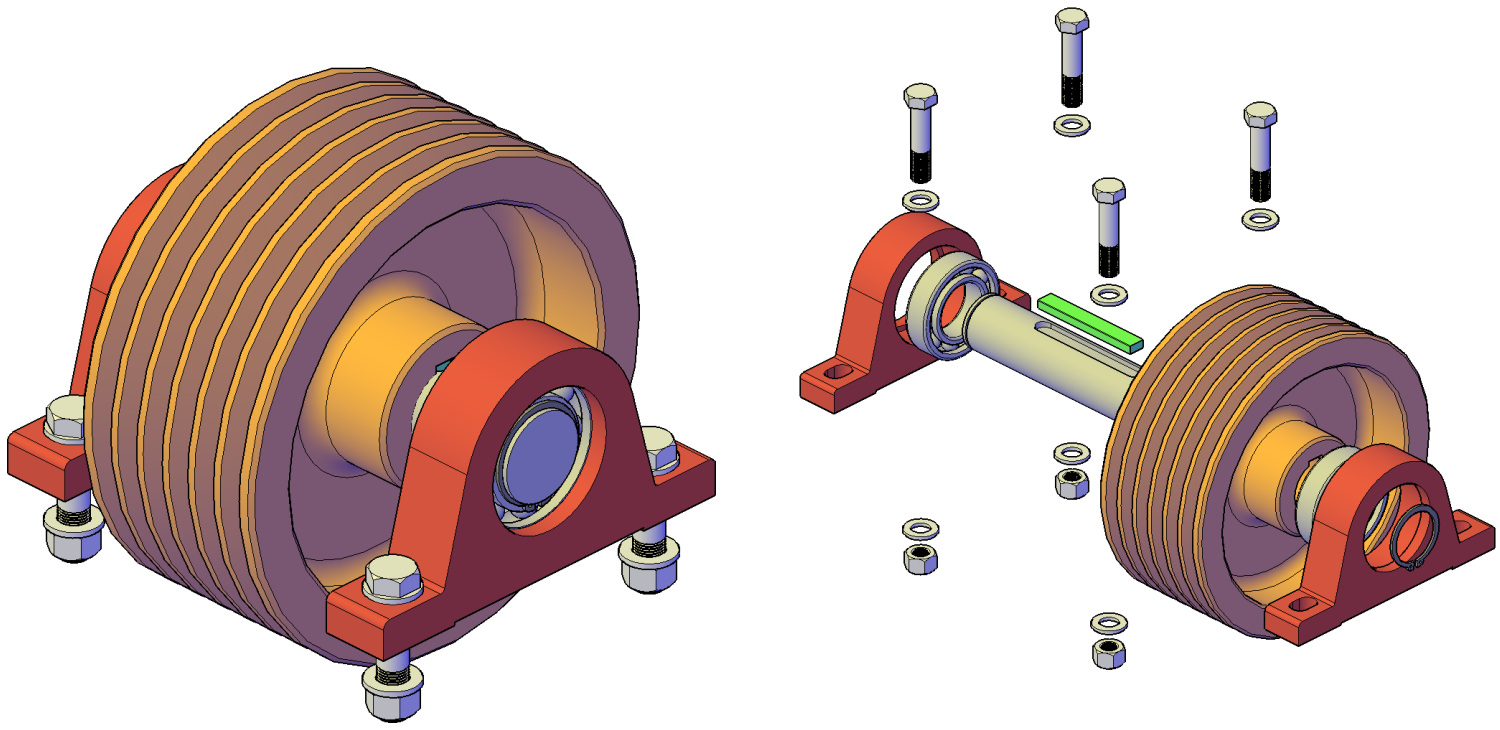


Results of Hierarchical Exploded View

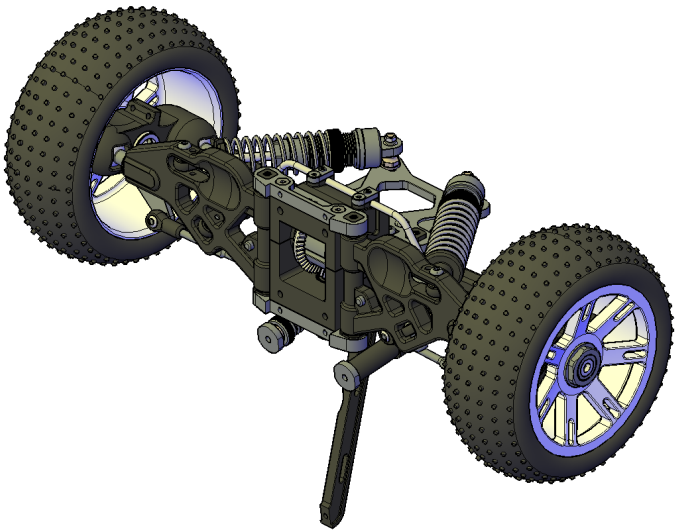


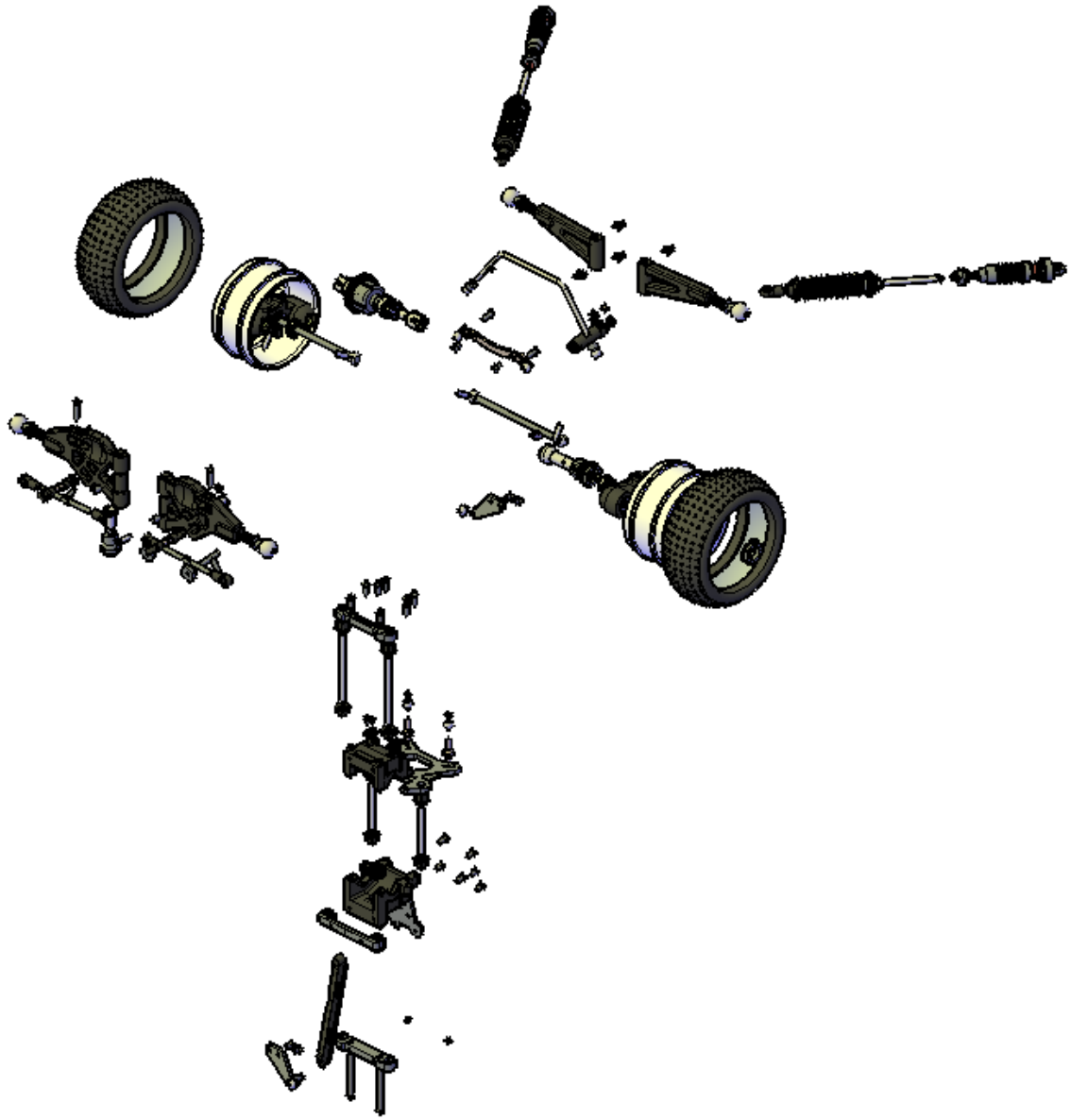


Results of Hierarchical Exploded View



Results of Hierarchical Exploded View





It took about 30 minutes

Comparison

Assembly	Number of parts	CPU time in seconds		
		Linear	Automatic EV	Hierarchical EV
Gem	73	54	—	245
Loader	61	7.65	27	45
2 Loaders	2×61	24.5	93	94
3 Loaders	3×61	50	200	143
V8	21	—	—	14
Difi	19	17	86	51
2 Difis	2×19	60	360	168
Wheel	24	—	6.4	5.5
2 Wheels	2×24	—	19.4	14.4
4 Wheels	4×24	—	60.5	28.3
8 Wheels	8×24	—	217	57.8
RC Buggy	217	—	—	1623

Section 6

Perspective

Perspective

- Improve `isBlockedBy` both in performance and accuracy
- Find the criterion to choose the natural direction
  - Parallelism
  - Continuity
  - Stability
  - Directionality
- Isometry
- Reinforcement Learning

Bibliography

W. Li, M. Agrawala, B. Curless, and D. Salesin, "Automated generation of interactive 3d exploded view diagrams", *ACM Trans. Graph.*, vol. 27, no. 3, 101:1–101:7, Aug. 2008, ISSN: 0730-0301. DOI: 10.1145/1360612.1360700. [Online]. Available: http://doi.acm.org/10.1145/1360612.1360700.

J. Yu and J. Zhang, "Hierarchical exploded view generation based on recursive assembly sequence planning",, vol. 93, pp. 1–22, Jun. 2017.