Automated Generation of Interactive 3D Exploded View Diagrams

Wilmot Li[1,3] Maneesh Agrawala[2] Brian Curless[1] David Salesin[1,3]
[1]University of Washington [2]University of California, Berkeley [3]Adobe Systems

Abstract

We present a system for creating and viewing interactive exploded views of complex 3D models. In our approach, a 3D input model is organized into an *explosion graph* that encodes how parts explode with respect to each other. We present an automatic method for computing explosion graphs that takes into account part hierarchies in the input models and handles common classes of interlocking parts. Our system also includes an interface that allows users to interactively explore our exploded views using both direct controls and higher-level interaction modes.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry & Object Modeling; I.3.8 [Computer Graphics]: Applications

Keywords: exploded view illustration, interactive, visualization

1 Introduction

Complex 3D objects, such as mechanical assemblies, electronic devices, and architectural environments, are typically composed of numerous parts. To convey the internal structure of such objects, illustrators often create exploded views in which parts are separated (or “exploded”) away from one another to reveal parts of interest. Well designed exploded views not only expose internal parts, they also convey the global structure of the depicted object and the local spatial relationships between parts. Furthermore, unlike other illustration techniques that reveal internal parts *in situ* by removing or de-emphasizing occluding geometry, such as cutaways and transparency, exploded views show the details of individual parts.

However, traditional static exploded views have several limitations that can make it difficult for viewers to browse the part structure of complex objects and focus on different subsets of parts. Since most exploded views expose all the parts in an object, they often suffer from excess visual clutter. As a result, the viewer may have to carefully inspect the entire illustration to locate parts of interest. Furthermore, parts that are close to one another may end up far apart when the object is fully exploded, making it difficult for the viewer to determine how parts of interest are positioned and oriented with respect to the rest of the model. Finally, static exploded views do not allow viewers to explore spatial relationships at different levels of detail. For instance, a viewer might first want to see how two subassemblies fit together before examining their constituent parts.

In this paper, we present a system for creating and viewing interactive 3D exploded views that allow users to explore the spatial relationships between specific parts of interest. In our approach, we automatically determine the order and directions in which parts can explode without violating blocking constraints (i.e., without passing through each other) and then use this information to implement high-level viewing tools that expand and collapse parts dynamically. For instance, the user can select target parts of interest from a list, and the system automatically generates an exploded view that exposes the targets without showing every other part in the object (see Figure 1). The user can then directly expand and collapse the exposed parts along their explosion directions to better see how they fit together.

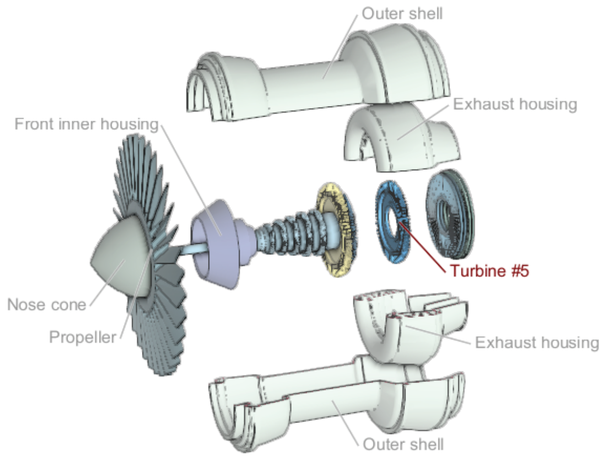


Figure 1 Exploded view diagram generated by our system. Our system instruments 3D models to enable interactive exploded views. This illustration of a turbine model was automatically computed to expose the user-selected target part labeled in red.

Our work makes several contributions. We present an automatic technique for organizing 3D models into layers of explodable parts that handles the most common classes of interlocking parts. We also introduce two algorithms for exposing user-selected target parts, one that explodes the necessary portions of the model in order to make the targets visible, and one that combines explosions with the dynamic cutaway views described by Li et al. [2007]. We also present several interactive viewing tools that allow the user to directly explore and browse our exploded views.

2 Related work

There is a large amount of existing work on visualizing the internal structure of complex 3D objects. Here, we focus on previous techniques that rearrange rather than remove geometry in order to expose parts of interest.

A number of digital illustration systems provide tools for creating exploded views [Adobe Inc. ; Agrawala et al. 2003; Driskill and Cohen 1995; Li et al. 2004; Rist et al. 1994]. A limitation of most of these systems is that the user must manually specify the explosion directions and blocking relationships for all parts in the model. A notable exception is the work of Agrawala et al. [2003], whose techniques for generating step-by-step assembly instructions automatically determine the order and directions in which parts can explode without violating blocking constraints. We present an automatic algorithm for computing exploded views that extends this work to take into account part hierarchies and to handle the most common cases in which parts interlock.

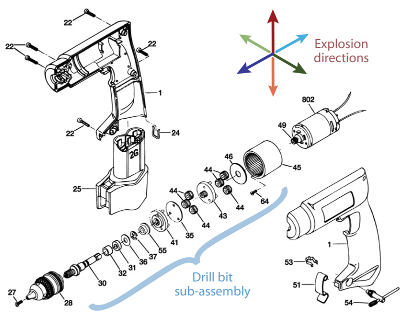


Figure 2 Explosion conventions. This illustration of a drill incorporates several explosion conventions described in Section 3.1. Image credit: Bosch Tools schematic

Most existing systems generate exploded views that are either meant to be viewed statically [Driskill and Cohen 1995; Rist et al. 1994] or provide limited interactive viewing controls (e.g., a single knob for expanding the entire exploded view [Agrawala et al. 2003]). Since a key goal of our system is to enable users to interactively explore complex 3D objects, we provide a richer set of interactions. The viewing interface for our system is similar to the work of Li et al. [2004], who present viewing tools that allow users to directly manipulate and search for parts. However, their techniques are designed for 2.5D illustrations, and thus do not automatically compute or maintain blocking constraints between parts.

Whereas traditional exploded views typically separate parts along linear explosion directions, researchers have developed visualization techniques that "explode" parts using other types of transformations and/or deformations. Some of these approaches peel away layers of parts using non-rigid deformations [Correa et al. 2006; McGuffin et al. 2003]. Others extend 2D fisheye techniques to expose and enlarge parts of interest in 3D scenes [Carpendale et al. 1997; LaMar et al. 2001; Raab and Rüger 1996; Wang et al. 2005]. Recent work by Bruckner and Gröller [2006] uses a force-based model to push occluding parts out of the way. Our work focuses on the challenges of generating more traditional exploded views.

3 Conventions from traditional illustration

The conventions described below were distilled from a large corpus of example illustrations taken from technical manuals, instructional texts on technical illustration, and educational books that focus on complex mechanical assemblies [Hoyt 1981; Platt and Biesty 1996; Dennison and Johnson 2003].

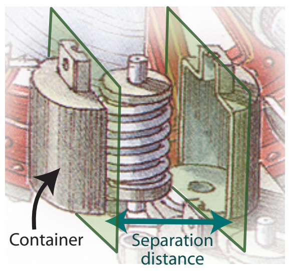
3.1 Explosion conventions

When creating exploded views, illustrators carefully choose the directions in which parts should be separated (*explosion directions*) and how far parts should be offset from each other based on the following factors. The example illustration in Figure 2 incorporates several of these conventions.

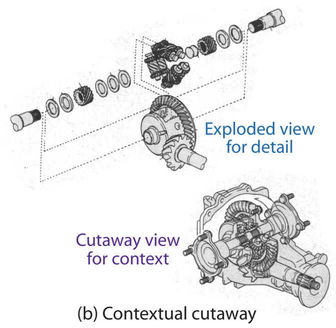
Blocking constraints. Parts are exploded away from each other in unblocked directions. The resulting arrangement of parts helps the viewer understand local blocking relationships and the relative positions of parts.

Visibility. The offsets between parts are chosen such that all the parts of interest are visible.

Compactness. Exploded views often minimize the distance parts are moved from their original positions to make it easier for the viewer to mentally reconstruct the model.



(a) Split container



(b) Contextual cutaway

Figure 3 Cutting conventions for exploded views. Image credits: (a) Stephen Biesty © Dorling Kindersley; (b) Bill Sherwood's Differential Page (*www.bilzilla.org*)

Canonical explosion directions. Many objects have a canonical coordinate frame that may be defined by a number of factors, including symmetry (as in Figure 2), real-world orientation, and domain-specific conventions. In most exploded views, parts are exploded only along these canonical axes. Restricting the number of explosion directions makes it easier for the viewer to interpret how each part in the exploded view has moved from its original position.

Part hierarchy. In many complex models, individual parts are grouped into *sub-assemblies* (i.e., collections of parts). To emphasize how parts are grouped, illustrators often separate higher-level sub-assemblies from each other before exploding them independently, as shown in Figure 2.

3.2 Cutting conventions in exploded views

Illustrators often incorporate cuts in exploded view diagrams. Here, we describe two common ways in which cuts are used.

Splitting containers. In many complex models, some internal parts are nested within container parts. To visualize such containment relationships, illustrators often split containers with a cutting plane through the centre of the part and then explode the two container segments away from each other to expose the contained parts (see Figure 3a). To emphasize that the segments originate from the same part, the orientation of the plane is chosen to minimize the distance the segments must be separated to make the internal parts visible.

Contextual cutaways. In some cases, a cutaway view is used to provide additional context for an exploded view. For example, in Figure 3b, the cutaway view (bottom) allows the viewer to see how the sub-assembly of interest is positioned and oriented with respect to surrounding structures, and the exploded view (top) exposes the sub-assembly's constituent parts.

4 Implementing exploded views

The minimum input to our system is a 3D solid model whose individual parts are represented as separate geometric objects. The following additional information may be specified to help the system generate higher quality exploded views.

Part hierarchy. If the input model is organized into a hierarchy of sub-assemblies, our system generates hierarchical exploded views that enable exploration at any level of the part hierarchy.

Explosion directions. By default, our system allows parts to explode only in directions parallel to the coordinate frame axes of the entire model. However, a different set of directions may be specified as part of the input. All of the example illustrations shown here were generated using the default explosion directions.

Cutaway instrumentation. If the input model is instrumented to enable cutaway views, as described by Li et al. [2007], our system

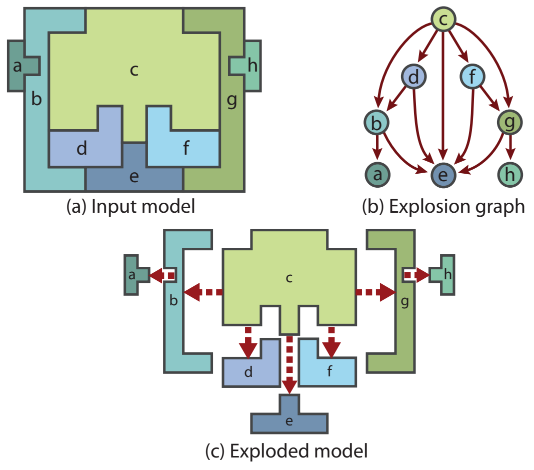


Figure 4 Explosion graph representation.

automatically combines exploded views with contextual cutaways to expose user-selected sub-assemblies.

In the remainder of this section, we describe a representation for 3D exploded views and then present an automatic technique for constructing this representation from the 3D input model.

4.1 Exploded view representation

To enable interactive exploded views, our system organizes parts into a directed acyclic *explosion graph*, as shown in Figure 4b. The structure of the graph defines the relative order in which parts can be exploded without violating blocking constraints. In particular, a part can explode as long as all of its descendants in the explosion graph have been moved out of the way. For each part p, the graph also stores an explosion direction and the current offset of p from its initial position. We define this initial position with respect to the largest of the direct parents of p, which encourages smaller parts to move together with larger parts. To expand and collapse different portions of the model, the system simply modifies the part offsets.

Since many of the computations described below need to know whether parts touch, block, or contain each other, our system computes auxiliary data structures that encode these low-level spatial relationships. Contact and blocking relationships are computed and stored in the manner described by Agrawala et al. [2003]. For a given part p, these data structures can be queried to determine all the parts that touch p and all the parts that block p from moving in each of the possible explosion directions. We use an approximate definition of containment that is computed as follows. For a pair of parts (p_1, p_2), our system checks whether the convex hull of p_2 is completely inside the convex hull of p_1 and if p_1 blocks p_2 from moving in each of the possible explosion directions. If so, p_1 is considered to contain p_2. The algorithms for computing contact, blocking, and containment relationships assume the input model is two-sided and that parts that are meant to fit together do not interfere with each other (i.e., overlap beyond a small tolerance).

4.2 Constructing the explosion graph

Our basic approach for computing explosion graphs is similar to the method of Agrawala et al. [2003] for determining assembly sequences. We first describe this algorithm before introducing two important extensions that allow our system to take into account part hierarchies and handle common classes of interlocking parts.

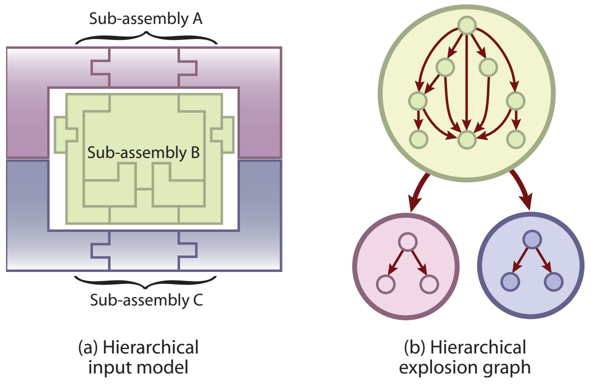

Sub-assembly A
Sub-assembly B
Sub-assembly C
(a) Hierarchical input model
(b) Hierarchical explosion graph

Figure 5 Hierarchical explosion graph.

Basic approach

To construct the explosion graph, we use an iterative algorithm that removes unblocked parts from the model, one at a time, and adds them to the graph. To begin, all model parts are inserted into a set S of active parts. At each iteration, the system determines the set of parts $P \subseteq S$ that are unblocked in at least one direction by any other active part. For each part $p \in P$, the system computes the minimum distance p would have to move (in one of its unblocked directions) to escape the bounding box of the active parts in contact with p. The part $p_i \in P$ with the minimum escape distance is added to the graph. An edge is added from every active part that touches p_i, and the direction used to compute the minimum escape distance is stored as the explosion direction for p_i. Finally, p_i is removed from S. The algorithm terminates when no unblocked parts can be removed from S.

Using part hierarchies

If the model has a part hierarchy, our system computes a nested collection of explosion graphs, as shown in Figure 5. This approach enables sub-assemblies at any level of the part hierarchy to expand and collapse independently. For each sub-assembly A, the system computes an explosion graph by treating all of the direct children of A in the part hierarchy (which may themselves be sub-assemblies) as atomic parts and then applying the algorithm described above.

Handling interlocked parts

In some cases, the parts in the active set may be interlocked such that no unblocked part can be removed. Here, we describe how our system handles two common classes of interlocking parts.

Splitting sub-assemblies
When computing the explosion graph for a hierarchical input model, the active set may contain interlocked sub-assemblies. In such cases, the system attempts to split interlocked sub-assemblies into smaller collections of parts (see Figure 6). Given an interlocked sub-assembly A, the system computes the largest *partial sub-assembly* (i.e., subset of parts in A) that can be separated from the remaining active parts and then removes this partial sub-assembly from the set of active parts. If there is more than one interlocked sub-assembly, the system computes the largest removable partial sub-assembly for each one. Amongst these computed partial sub-assemblies, the smallest one is removed from the set of active parts.

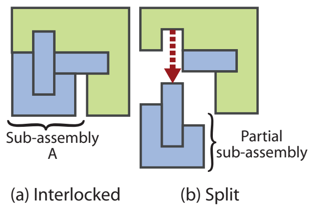


Figure 6 Splitting sub-assembly A.

Splitting containers

If any of the interlocked parts is an atomic container part whose only blockers are contained parts, the system splits the container into two segments that are then removed from the set of active parts, as shown in Figure 7. To split a container c, the system selects one of the candidate explosion directions and then splits c into two segments c_1 and c_2 with a cutting plane that passes through the bounding box centre of c and whose normal is parallel to the chosen explosion direction. The explosion direction is determined in a view-dependent manner. The system explodes the set of contained parts P and then, for each candidate direction, measures how far c_1 and c_2 would have to separate in order to completely disocclude and escape the 3D bounding box of P (see Figure 7b). In accordance with the cutting conventions described in Section 3.2, the container c is split in the direction that requires the smallest separation distance.

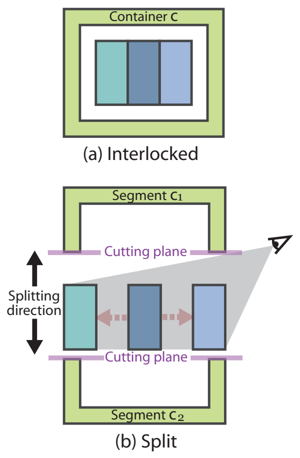


Figure 7 Splitting container c.

If some of the parts in P are themselves containers, the system emphasizes their concentric containment relationships by considering only explosion directions where the bounding boxes of the nested containers remain inside the exploded bounding box of c. If none of the splitting directions satisfy this constraint, the system chooses the splitting direction that causes the smallest total volume of nested container bounding boxes to extend beyond the exploded bounding box of c.

4.3 Precomputation

Since the viewing direction can influence how container parts are split, explosion graphs may be view-dependent. Recomputing these data structures on the fly as the viewpoint changes can cause some lag in the viewing interface. Instead, our system precomputes explosion graphs from the 26 viewpoints that correspond to the faces, edges and corners of an axis-aligned cube that is centered at the model's bounding box center and is large enough to ensure that the entire model is visible from each viewpoint. At viewing time, the system automatically switches to the precomputed explosion graph closest to the current viewpoint.

5 Viewing interactive exploded views

Once the explosion graph is computed, the model can be explored in our viewing interface, which provides both direct controls and higher-level interaction modes to help users find parts of interest and explore specific portions of the model.

5.1 Animated expand/collapse

Our system allows the user to expand or collapse the entire exploded view with a single click. Each part is animated to its fully exploded or collapsed position by updating its current offset. To ensure that parts do not violate blocking constraints during the animation, the system expands parts in reverse topological order (i.e., outermost to innermost) with respect to the explosion graph. In other words, the descendants of each part are expanded before the part itself. For hierarchical models, higher-level (i.e., larger) sub-assemblies are expanded before lower-level sub-assemblies. The system collapses parts in the opposite order.

5.2 Direct manipulation

The system also supports the direct manipulation of parts. As the user drags a part p, the system slides p along its explosion direction and updates the current offset of p. If the user drags p past its fully exploded or collapsed position, the system propagates the offset through the explosion ancestors of p until it encounters a part with a different explosion direction. Propagating offsets in this manner allows the user to expand or collapse an entire collection of parts, just by dragging a single part.

This type of constrained direct manipulation for exploded views was introduced in the image-based system of Li et al. [2004]. However, that approach does not automatically compute and enforce blocking constraints. In our system, blocking constraints are maintained in real time during direct manipulation. As the user drags part p, the system checks for blocking parts amongst the descendants of p in the explosion graph and stops p from moving if such parts are found. A single click causes the blocking parts to move out of the way, which allows the user to continue dragging.

5.3 Riffling

The viewing interface also provides a *riffling* mode, in which parts are exploded away from adjacent portions of the model as the user hovers over them with the mouse. When the mouse moves away, the part that was beneath the mouse returns to its initial position. If the user clicks, the selected part remains separated as the mouse moves away. Although similar in feel to existing 3D fisheye viewing techniques [LaMar et al. 2001; Sonnet et al. 2004], our riffling interaction restricts spatial distortions to the computed explosion directions. By riffling through the model, the user can quickly isolate parts or sub-assemblies and see how various portions of the model can expand without actually dragging on a part.

5.4 Automatically exposing target parts

In addition to the direct controls described above, the viewing system provides a high-level interface for generating exploded views that expose user-selected target parts. The user just chooses the targets from a list of parts and the system automatically generates a labeled exploded view illustration. Parts are smoothly animated to their new positions to help the user see which portions of the model have been expanded and collapsed. The text labels are arranged using the approach described by Ali et al. [2005].

We describe two different techniques for generating illustrations. By default, the system expands specific portions of the model to expose the target parts. If the model has been instrumented for cutaways (as described by Li et al. [2007]), the system can also generate illustrations that combine explosions with contextual cutaways.

Exposing target parts with explosions

For non-hierarchical models, the algorithm works as follows. Given a set of target parts T, the system visits each part p in topological order with respect to the explosion graph and moves p if necessary to ensure that no visited target part is occluded by any other visited part. That is, p is moved to meet the following two conditions:

1. p does not occlude any previously visited target parts.
2. if $p \in T$, p is not occluded by any visited part.

To visually isolate target parts from surrounding parts, the algorithm moves p to meet two additional conditions that ensure each target is separated from its touching parts, even if those touching parts do not actually occlude the target:

3. p is not occluded by any visited target part that touches p.
4. if $p \in T$, p does not occlude any visited part that touches p.

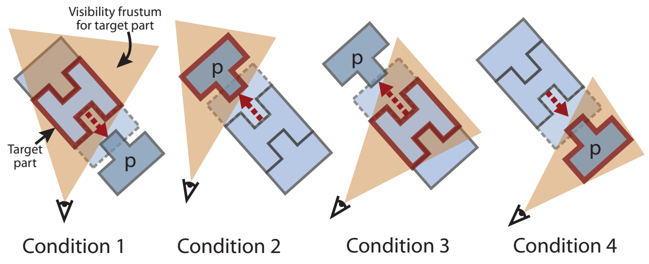


Figure 8 Conditions for moving part p. For each condition, the target part is outlined in red. The orange visibility frusta show how unwanted occlusions have been eliminated in each case.

To satisfy these conditions, the system performs the relevant occlusion tests and if necessary, moves p the minimum distance along its explosion direction such that all unwanted occlusions are eliminated (see Figure 8). To successfully eliminate unwanted occlusions, the explosion direction of p must not be exactly parallel to the viewing direction. If it is parallel, the system informs the user that one of the targets cannot be exposed from this viewpoint; in practice, such failure cases rarely arise. If p is moved, its explosion graph descendants are also moved out of the way so that no blocking constraints are violated. Since the position of a part only depends on the positions of its explosion graph ancestors, visited targets are guaranteed to remain visible with respect to visited parts after each part is processed. Thus, once every part has been processed, the resulting exploded view will have no occluded targets.

For hierarchical models, the algorithm starts by processing the highest level sub-assembly in the part hierarchy. Atomic parts and sub-assemblies that do not contain any target parts are processed as described above to eliminate target occlusions. However, when the algorithm encounters a sub-assembly A that contains one or more target parts, the algorithm recursively processes the parts within A to expose these targets. Once this recursive procedure returns, the system checks whether A (in its new configuration) violates blocking constraints with respect to any visited parts or occludes any visited targets not contained in A. If so, the algorithm iteratively increases the current offset of A and then repeats the recursive computation for A until no blocking constraints are violated and all visited targets are visible (see Figure 9). At each iteration, the current offset of A is increased by one percent of the bounding box diagonal for the entire model. The exploded views shown in Figures 1, 10a, and 10b were generated using this algorithm.

Exposing target sub-assemblies with cutaways and explosions

If the input model is instrumented for cutaways and the user selects an entire sub-assembly A as a target, the system first generates a cutaway view that exposes A in context and then explodes A away from the rest of the model through the cutaway hole. Finally, the system explodes A itself to expose its constituent parts (see Figure 11). To generate the cutaway, the system first chooses an explosion direction for A. Given the viewing direction v, the system chooses the explosion direction d that allows A to escape the model's bounding box as quickly as possible and satisfies the constraint $d \cdot v < 0$. Using the method of Li et al. [2007], the system creates a cutaway that is large enough to allow A to explode away from the rest of the model in direction d.

6 Results

We have used our system to generate exploded views of several 3D models, as shown in Figures 1, 10, 11, 12, and 13. The iPod model was downloaded from TurboSquid (*www.turbosquid.com*), the arm dataset is from a commercially available model of human anatomy created by Zygote Media (*www.zygote.com*), and the rest

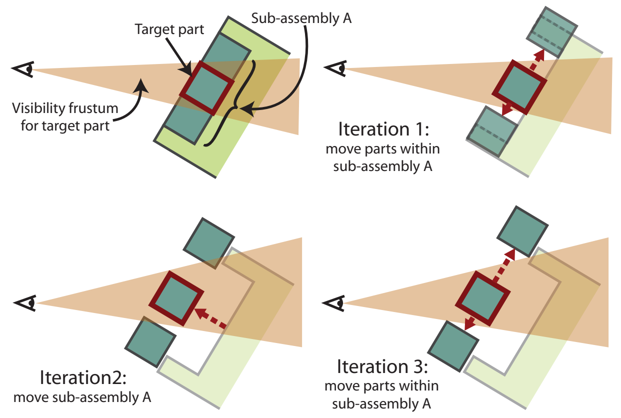

Target part
Sub-assembly A
Visibility frustum for target part
Iteration 1: move parts within sub-assembly A
Iteration2: move sub-assembly A
Iteration 3: move parts within sub-assembly A

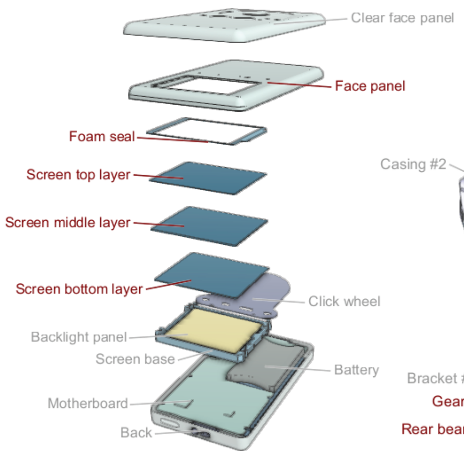
Figure 9 Exposing a target part within a hierarchical model. To expose the target part within sub-assembly A, the algorithm iteratively removes target occlusions within A (iteration 1) and moves A itself to enforce blocking constraints (iteration 2). When the algorithm converges at iteration 3, the target is visible and all blocking constraints are satisfied.

of the datasets were obtained from a public repository for CAD models. We made a few modifications to some of these models before loading them into our system. To satisfy our input assumptions, we scaled two of the transmission parts slightly to eliminate interferences with adjacent parts. For the turbine, disk brake, and grip mechanism examples, we omitted some of the original parts to reduce the preprocessing time required to compute contact and blocking relationships. The iPod model was poorly segmented, so we manually grouped some of the geometry into parts before loading it into our system. We created part hierarchies for the disk brake and transmission models based on the simple part groupings that came with the models. For the other CAD datasets, we created hierarchies based on the spatial organization and material properties of the parts. The arm model has no part hierarchy.

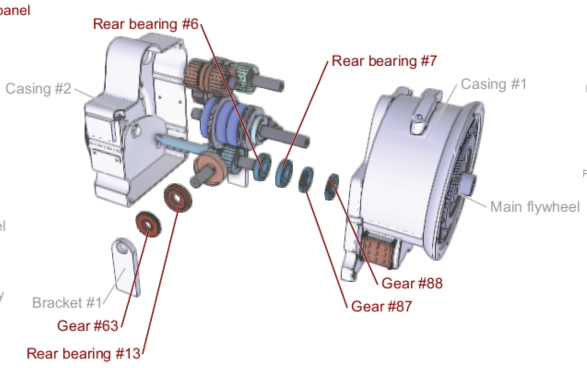
To generate the exploded view of the turbine shown in Figure 1, the system automatically determined how to split two container parts: the outer shell and the exhaust housing. The exploded views shown in Figures 1, 11, 10a, and 10b were generated automatically to expose user-specified target parts. These illustrations clearly show the parts of interest without exploding unnecessary portions of the model. In addition, Figure 11 shows how a contextual cutaway view helps convey the position and orientation of the exploded sub-assembly with respect to the rest of the model.

Although exploded views are typically used to illustrate manufactured objects, we also tested our system with a musculoskeletal model of a human arm. Since many of the muscles in the arm twist around each other where they attach to bones, we manually sectioned off part of the arm where the muscles are less intertwined and then used this portion of the dataset as the input to our system. To emphasize how the muscles are layered from the outside to the inside of the arm, we also restricted the system to use a single explosion direction. From this input, our system automatically computed the exploded view shown in Figure 12.

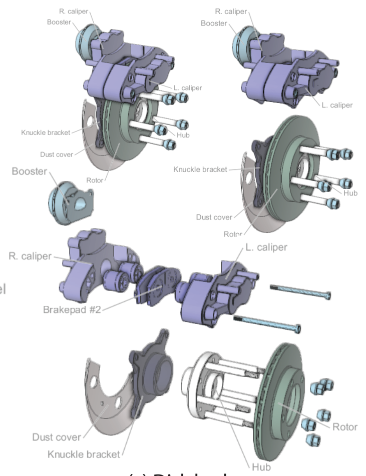
Table 1 reports the number of parts and precomputation time for each dataset. In general, computing contact and blocking relationships dominates the total precomputation cost. We do not report the cost of computing containment relationships because containment tests are performed lazily only when the system encounters interlocked parts during explosion graph construction. Since the turbine is the only dataset that includes container parts, its explosion graph cost includes the time required for containment tests.



(a) iPod



(b) Transmission



(c) Disk brake

Figure 10 Exploded views generated by our system. The illustrations of the iPod (a) and transmission (b) were automatically generated to expose the user-selected target parts labeled in red. The sequence of images on the right shows the disk brake model exploding in stages (c).

Model	N_{parts}	$T_{contact}$	T_{block}	T_{egraph}	T_{total}
Disk brake	18	440s	195s	0.016s	635s
iPod	19	215s	20s	0.016s	235s
Grip mechanism	20	150s	50s	0.016s	200s
Arm	22	280s	40s	0.016s	320s
Turbine	26	380s	200s	430s	1010s
Carburetor	42	135s	45s	0.16s	180s
Transmission	55	885s	835s	2.5s	1778s

Table 1 Precomputation statistics. For each model, we report the number of parts N_{parts} and the time (in seconds) required to compute contacts $T_{contact}$, blocking relationships T_{block}, and explosion graphs T_{egraph}. The total precomputation time T_{total} is listed on the right. All timings were performed on a MacBook Pro with a 2.6 GHz Intel Core 2 Duo, 4GB of memory, and an nVidia GeForce 8600M GT graphics card.

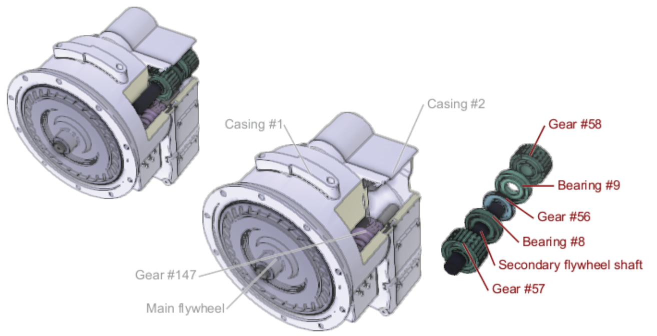


Figure 11 Exploded view with contextual cutaway. To expose the user-selected sub-assembly, the system first generates a cutaway view (left) and then explodes the sub-assembly through the cutaway hole (right).

7 Discussion

Although in general our system produces effective exploded view diagrams for a large class of 3D models, our approach does have some limitations. As mentioned earlier, the contact, blocking, and containment computations assume the input model is two-sided and that parts fit together without interferences. In addition, our algorithm for constructing explosion graphs assumes that all parts can be separated via rigid linear translations. Thus, the system is not able to separate atomic parts with certain complex interlocking relationships (e.g., a screw modeled with realistic thread geometry would have to rotate to be removed). Although not all 3D models adhere to these input assumptions, many CAD models are specifically designed to fit together properly without interferences. Furthermore, for performance reasons, CAD programs typically include part libraries with stylized models of standard screws, bolts and other fasteners that do not have realistic thread geometry.

Figure 13 illustrates two situations in which our approach can generate less successful results. Since the blocking algorithm that we use analyzes the normals of contact faces to determine blocking directions, the computation can be sensitive to noisy or irregular surface geometry where parts touch each other. For example, in Figure 13a, the top shaft explodes sideways rather than upwards from the top spring because the complexity of the spring geometry causes the system to compute the wrong blocking relationship. Figure 13b shows a case in which splitting a non-container part could help clarify spatial relationships. Although the illustration successfully exposes the target parts, it is not obvious how the exposed parts fit together within the long hollow body. Splitting the body lengthwise and then exploding the two halves away from each other would likely provide better spatial context for the internal parts. However, since the parts are free to slide out of the body, the system does not consider the body to be a container part (and thus, does not incorporate any cuts).

Finally, although we have shown that our system can generate effective visualizations of models with up to roughly 50 parts, there are some challenges involved in scaling our approach to handle significantly more complex models. First, exposing an internal part might require exploding many blocking parts, which could result in visual clutter. We believe our approach could be extended with level-of-detail controls to reduce the amount of clutter due to exploded parts. In addition, the cost of computing contact, blocking, and containment relationships puts a practical limit on the complexity of the input geometry. Our current implementation incorporates simple spatial data structures to accelerate these computations, but for highly complex input models, more efficient algorithms would likely be necessary to avoid prohibitively large precomputation times.

8 Conclusions and future work

In this paper, we have presented techniques for creating and viewing interactive exploded view illustrations of 3D models composed of many distinct parts. Our contributions include an automatic method for decomposing models into explodable layers and algorithms for generating dynamic exploded views that expose user-selected target parts. Our results demonstrate that our approach can be used to create effective interactive exploded views for a variety of models.

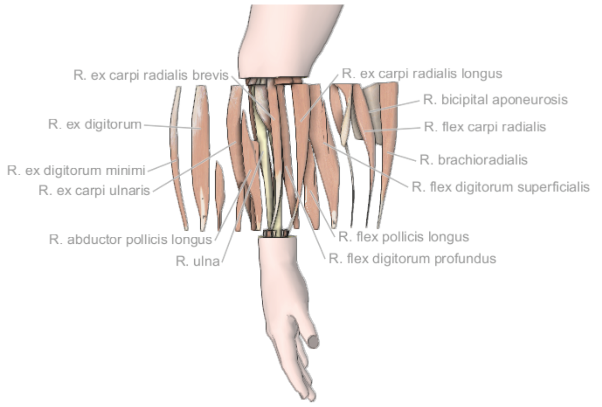


Figure 12 Exploded view of arm. To create this visualization, we sectioned off a portion of the arm to explode. Within this portion, our system automatically computed the layering relationships between the muscles.

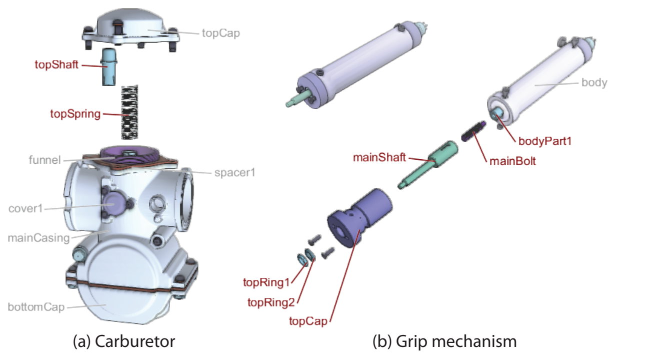


Figure 13 Less successful results. Due to the complexity of the spring geometry, the system computes the wrong blocking relationship for the top shaft in the carburetor model (a). In the exploded view of the grip mechanism (b), it may be hard for the viewer to understand how the highlighted target parts fit together inside of the long hollow body.

We conclude by mentioning a few areas for future work:

Inferring parts to expose. Our system allows users to directly specify target parts to expose. In some cases, showing additional parts can provide more context for the specific parts of interest. It would be interesting to explore techniques that automatically infer which additional parts to expose based on the user-selected targets.

Continuous measure of blocking. It is difficult to detect the correct blocking relationships for 3D models that contain interfering parts. One potential approach for handling such models would be to construct exploded views based on some continuous measure of the "amount" of blocking between parts.

Automatic guidelines. Guidelines can be useful for clarifying how exploded parts fit back together. Although some previous work [Agrawala et al. 2003] has been done on computing guideline placement automatically, further investigation is required to identify and implement all of the conventions that illustrators use to create effective guidelines.

Acknowledgments

The authors want to thank Sharon Lin for her work on the label layout component of the system. We also thank François Chrétien, Doug Johnson, and Dave Kasik for their feedback on the use of CAD technology in industry. This work was supported by NSF grants CCF-0643552 and EIA-0321235, the Alfred P. Sloan Foundation, the University of Washington Animation Research Labs, the Washington Research Foundation, Adobe, Microsoft, and Pixar.

References

ADOBE INC. Acrobat 3D.

AGRAWALA, M., PHAN, D., HEISER, J., HAYMAKER, J., KLINGNER, J., HANRAHAN, P., AND TVERSKY, B. 2003. Designing effective step-by-step assembly instructions. *ACM Transactions on Graphics 22*, 3 (July), 828–837.

ALI, K., HARTMANN, K., AND STROTHOTTE, T. 2005. Label layout for interactive 3D illustrations. In *WCSG Journal*, 1–8.

BRUCKNER, S., AND GROLLER, M. 2006. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct.), 1077–1084.

CARPENDALE, M. S. T., COWPERTHWAITE, D. J., AND FRACCHIA, F. D. 1997. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications: Special Issue on Information Visualization 17*, 4, 42–51.

CORREA, C., SILVER, D., AND CHEN, M. 2006. Feature aligned volume manipulation for illustration and visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (Sept./Oct.), 1069–1076.

DENNISON, J. A., AND JOHNSON, C. D. 2003. *Technical Illustration: Techniques and Applications*. Goodheart-Wilcox.

DRISKILL, E., AND COHEN, E. 1995. Interactive design, analysis and illustration of assemblies. In *Proceedings of the Symposium on Interactive 3D Graphics*.

HOYT, W. A. 1981. *Complete Car Care Manual*. Reader's Digest.

LAMAR, E., HAMANN, B., AND JOY, K. I. 2001. A magnification lens for interactive volume visualization. In *9th Pacific Conference on Computer Graphics and Applications*, 223–232.

LI, W., AGRAWALA, M., AND SALESIN, D. H. 2004. Interactive image-based exploded view diagrams. In *Proceedings of Graphics Interface 04*.

LI, W., RITTER, L., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2007. Interactive cutaway illustrations of complex 3D models. *ACM Transactions on Graphics 26*, 3 (July), 31:1–31:11.

MCGUFFIN, M. J., TANCAU, L., AND BALAKRISHNAN, R. 2003. Using deformations for browsing volumetric data. In *Proceedings of IEEE Visualization 2003*, 401–408.

PLATT, R., AND BIESTY, S. 1996. *Stephen Biesty's Incredible Explosions*. DK Children.

RAAB, A., AND RÜGER, M. 1996. 3D-ZOOM: interactive visualization of structures and relations in complex graphics. In *3D image analysis and synthesis*, 87–93.

RIST, R., KRÜGER, A., SCHNEIDER, G., AND ZIMMERMANN, D. 1994. AWI: A workbench for semi-automated illustration design. In *Proceedings of Advanced Visual Interfaces 94*.

SONNET, H., CARPENDALE, S., AND STROTHOTTE, T. 2004. Integrating expanding annotations with a 3D explosion probe. In *Proceedings of ACM AVI 2004*, 63–70.

WANG, L., ZHAO, Y., MUELLER, K., AND KAUFMAN, A. E. 2005. The magic volume lens: An interactive focus+context technique for volume rendering. In *Proceedings of IEEE Visualization 2005*, 367–374.