Compact Explosion Diagrams

Markus Tatzgern*
Graz University of Technology

Denis Kalkofen†
Graz University of Technology

Dieter Schmalstieg‡
Graz University of Technology

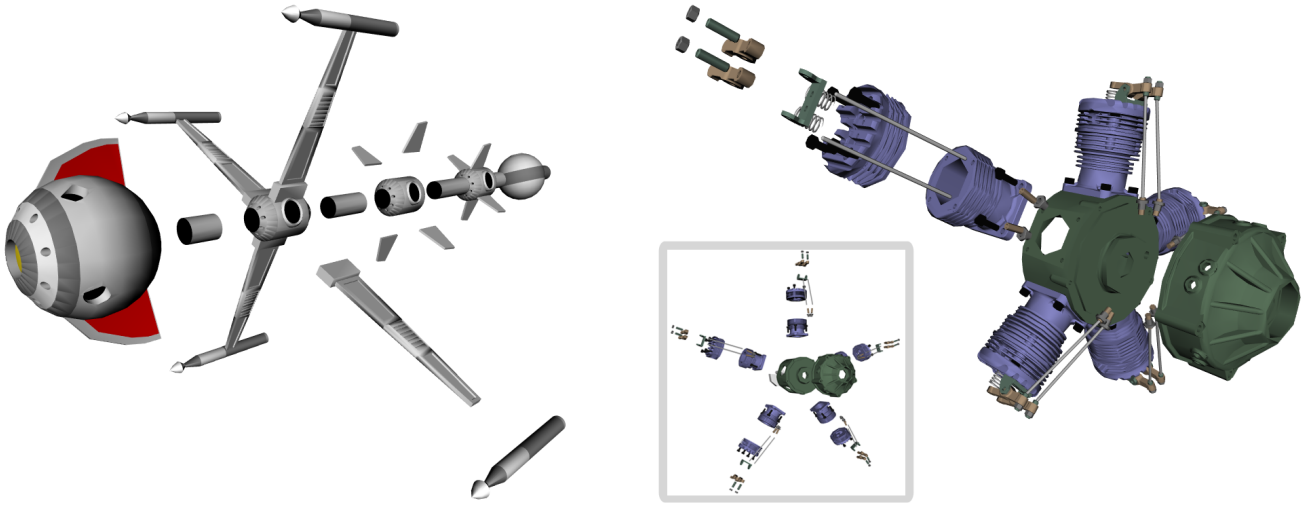


Figure 1: *Compact Explosion Diagrams. The entire assemblies are demonstrated by a set of representative exploded views. Our system automatically detects recurring subassemblies before it computes the one which represents the composition best.*

Abstract

This paper presents a system to automatically generate compact explosion diagrams. Inspired by handmade illustrations, our approach reduces the complexity of an explosion diagram by rendering an exploded view only for a subset of the assemblies of an object. However, the exploded views are chosen so that they allow inference of the remaining unexploded assemblies of the entire 3D model. In particular, our approach demonstrates the assembly of a set of identical groups of parts, by presenting an exploded view only for a single representative. In order to identify the representatives, our system automatically searches for recurring subassemblies. It selects representatives depending on a quality evaluation of their potential exploded view. Our system takes into account visibility information of both the exploded view of a potential representative as well as visibility information of the remaining unexploded assemblies. This allows rendering a balanced compact explosion diagram, consisting of a clear presentation of the exploded representatives as well as the unexploded remaining assemblies. Since representatives may interfere with one another, our system furthermore optimizes combinations of representatives. Throughout this paper we show a number of examples, which have all been rendered from unmodified 3D CAD models.

Keywords: Spatial layout techniques, illustrative rendering.

1 Introduction

Explosion diagrams provide a powerful presentation technique to enable comprehensible explorations of three dimensional objects. While other illustrative exploration techniques, such as cutaways [Li et al. 2007] or ghostings [Kalkofen et al. 2009a] remove parts from the presentation, explosion diagrams present all parts entirely opaquely and within full detail, by displacing the elements of an object. The displacements are carefully designed to encode the assembly of the object. While artists have been trained to intuitively choose comprehensible arrangements of parts, computer graphics scientists have been investigating algorithms to compute the layout of an explosion diagram automatically. For example, the current state of the art algorithms automatically define relations between parts, which are subsequently used to control their displacement [Agrawala et al. 2003; Li et al. 2008].

However, automatically generated explosion diagrams of complex

*e-mail: tatzgern@icg.tugraz.at
†e-mail: kalkofen@icg.tugraz.at
‡e-mail: schmalstieg@icg.tugraz.at

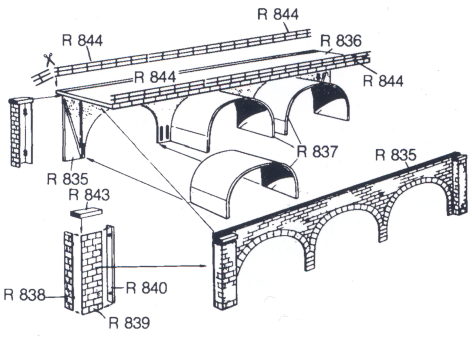


Figure 2: *Handmade compact explosion diagram [Mijksenaar and Westendorp 1999] . The assembly of the entire model is presented by a set of representative exploded views only.*

objects can easily suffer from cluttered layouts. While artists intuitively decide which parts of a complex explosion diagram are really necessary, computer graphics applications had to resort to user interaction to control the complexity at runtime [Li et al. 2008]. Such interaction requires a certain effort, which increases with the complexity of the 3D model. In addition, traditional media, such as textbooks, do not allow interaction with the presentation at all. Illustrations targeted for non-interactive media are still required to present the entire assembly of an object.

1.1 Contribution

In this paper, we present a system which is able to automatically reduce the complexity of a still explosion diagram. Inspired by handmade illustrations, such as demonstrated in Figure 2, we reduced the complexity of an explosion diagram by rendering an exploded view only for a subset of the assemblies of an object. The exploded views are chosen so that they allow inference of the remaining unexploded assemblies of the entire 3D model. Note how the illustrator of Figure 2 uses a selective displacement multiple times to render a more compact explosion layout.

The presented approach automatically mimics this layout technique. Our renderings demonstrate the assembly of a set of identical groups of parts by presenting an exploded view only for a single representative. In order to identify the representatives, our system automatically searches for frequent subassemblies. We select representatives depending on a quality evaluation of its potential exploded view, including size and explosion directions. Moreover, our system takes into account visibility information of the remaining unexploded assemblies. This allows rendering a balanced compact explosion diagram consisting of a clear presentation of both the exploded representatives and the unexploded remaining assemblies. Since representatives may interfere with one another, our system furthermore optimizes combinations of representatives using the approach of threshold acceptance [Dueck and Scheuer 1990].

1.2 System Overview

The presented approach reduces the visual complexity of an explosion diagram by reducing the number of exploded assemblies to a set of representatives. To select those which will be disassembled, our system first searches for assemblies which appear multiple times in the model. Subsequently, representative assemblies will be selected to demonstrate the assembly. Finally a pre-computed explosion layout is used to disassemble the selected representatives.

The architecture of our system is presented in Figure 3. It consists of three components. An initial explosion layout (Figure 3(A)) is used to present to render an exploded view of a set of representatives. An optimized combination of representatives is computed (Figure 3(C)) based on a previous detection of groups of similar assemblies (Figure 3(B)).

In the remainder of this paper we present each of these components in detail. Section 3 discusses our approach to compute an explosion layout. Note, since a representative exploded view has to demonstrate all other disassemblies of the same type, the algorithm to compute an explosion layout has to compute similar exploded views of similar assemblies. Section 4 presents our algorithm to find similar assemblies before Section 5 describes our approach to select an optimal combination of representative disassemblies, even in different levels of hierarchical groups.

The 3d models used in this paper were downloaded from public repositories or created from scratch. The spaceship shown in Figure 1 was downloaded from The 3D Studio (*www.the3dstudio.com*). The radial engine and the plane were obtained from Google 3D Warehouse (*sketchup.google.com/3dwarehouse*).

2 Related Work

Over the past 15 years, computer graphic researchers have investigated a number of different methods to automate the generation of explosion diagrams. These methods create explosion diagrams from many different kinds of data, ranging from 3D-CAD data [Rist et al. 1994], triangle soups [Niederauer et al. 2003] and volumetric data [Bruckner and Gröller 2006] to 2D image data [Li et al. 2004]. In addition, a number of different approaches have been presented to automatically compute the explosion's layout. Distortion techniques as presented by Raab [Raab and Rüger 1996] scale occluding parts. Force based techniques as presented by Sonnet [Sonnet et al. 2004] and Bruckner [Bruckner and Gröller 2006] use a set of interactively applied repelling and attracting forces, which allow defining the directions and distances for offset parts. Agrawala et al. [Agrawala et al. 2003] and Li et al. [Li et al. 2008] use spatial blocking information between parts as well as a size analysis to automatically derive the relations and directions.

To control the visual complexity of an explosion diagram, the existing approaches mainly provide interactive techniques. For example, Sonnet et al. [Sonnet et al. 2004] presented an interactive system which moves parts of an object out of the 3D volume of an explosion probe. Bruckner et al. [Bruckner and Gröller 2006] interactively define the amount and the relationships between forces to control the distances, direction and relative movements of parts. Li et al. [Li et al. 2008] presented techniques, such as dragging or riffling of parts to interactively explore a pre-computed explosion diagram, starting from a completely unexploded presentation.

Even though research on rendering of explosion diagrams has often focused on interactive systems, a few others have investigated an automatic search of groups of parts to simplify the explosion layout. Thus, the works closest to our approach are the systems of Ruiz et al. [Ruiz et al. 2008], Kalkofen et al. [Kalkofen et al. 2009b], Agrawala et al.[Agrawala et al. 2003] and Niederauer et al. [Niederauer et al. 2003]. Niederauer et al. [Niederauer et al. 2003] attempt to explode the floors of a building searching for those triangles which group up to a floor. Since different floors are usually offset at a certain distance and oriented similarly, Niederauer was able to find groups of triangles by applying a statistical analysis of their locations and orientations. Ruiz et al. [Ruiz et al. 2008] define the thicknesses of parallel slabs of a volume, based on a similarity measure between neighboring slabs. The similarity values are computed using mutual information computations. While the former

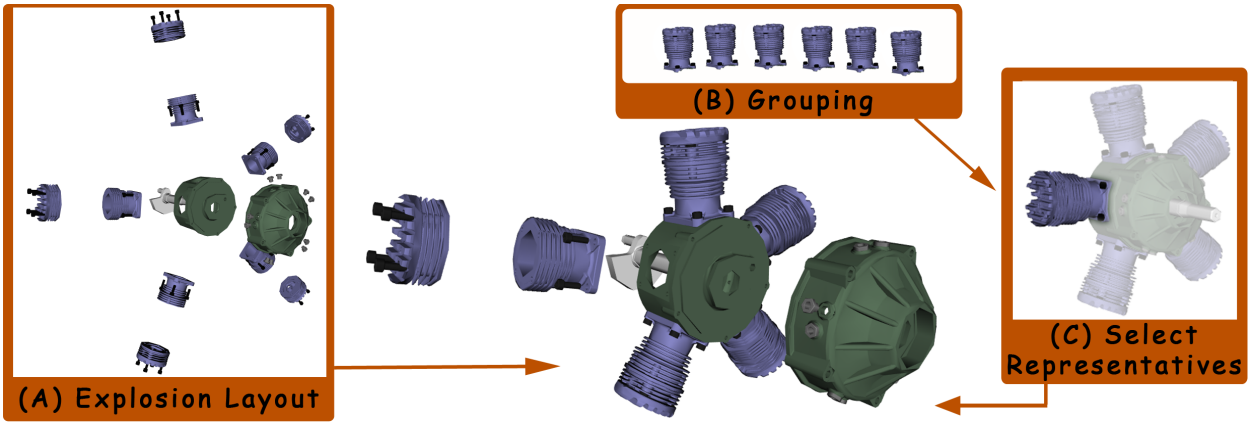


Figure 3: *System Architecture. Our system consists of three different modules which affect the rendering of compact explosion diagrams. By supplying a 3D CAD model, it automatically computes an initial explosion layout (A), it finds groups of equal parts (B) and it selects a representative (C) before it initiates the rendering.*

approach only performed well on structures similar to buildings, the latter is optimized for volumetric data.

Explosions of groups of parts of a 3D CAD model have been presented by Kalkofen et al. [Kalkofen et al. 2009b], Agrawala and later Li et al. [Agrawala et al. 2003; Li et al. 2008]. While Agrawala and Li manually annotated their models with group information, Kalkofen and his colleagues automatically group elements based on a selected focus element, which they aim to uncover. In a complete AND/OR-Graph data structure, they search for the largest groups of parts which can be displaced from the subassembly containing the object of interest. By recursively applying this search strategy on the AND/OR-Graph data structure, their approach is able to compute a Focus and Context explosion layout with an uncovered object of interest and a minimal amount of contextual groups.

3 Initial Explosion Layouts

The layout of an explosion diagram depends on the direction and the distance chosen for each part, to set it apart from its initial position. To reduce the mental load to reassemble an exploded object, explosion directions often follow mounting directions, therefore collisions between displaced parts are avoided. Explosion diagrams implement this feature by introducing relations between the parts of an assembly. For example, each screw in Figure 3 moves relative to the purple cylinder which it fastens. By displacing one of the purple cylinders, the corresponding black screws will be displaced implicitly.

The relationships between parts of an explosion diagram also allow parts to follow related parts. This enables a part to move relative to its initial location in the assembly, which also reduces the number of mental transformations to reassemble the object. For example, note how the grey bolts follow the green gearbox in the explosion diagram in Figure 3. However, it is often not obvious which part best represents the initial location of another part. For example, while the initial locations of the black screws in Figure 3 are clearly defined by the holes of the engine they fasten, the initial location of the wheels in Figure 4 is surrounded by a number of parts. As demonstrated in Figure 4(a), the wheels in front of the car may follow the blue steering gear. This will result in a translation along the up-vector of the car, before the wheels explode along the x-directions of the model's coordinate system. In contrast, the explosion diagram in Figure 4(b) uses a relation between the wheels in the front of the car and the green base-plate. This results in a displacement of the wheels without a translation along the up-vector of the coordinate system. The same behavior appears for a stack of parts in the back of the car. Since in Figure 4(a) the parent of the stack follows the red seat of the car, all the parts between the wheels and the seat have been moved along the up-vector before they have been separated from each other. In contrast, the explosion diagram in Figure 4(b) uses a relationship between the parent of the stack and the green base-plate of the car, which reduces the number of translations of all the elements in the stack.

3.1 Disassembly Sequence, Part Relations and Explosion Directions

We define relations between parts by computing a disassembly sequence. A relationship is set up for each exploded part and the biggest part in the remaining assembly it has contact with. To avoid

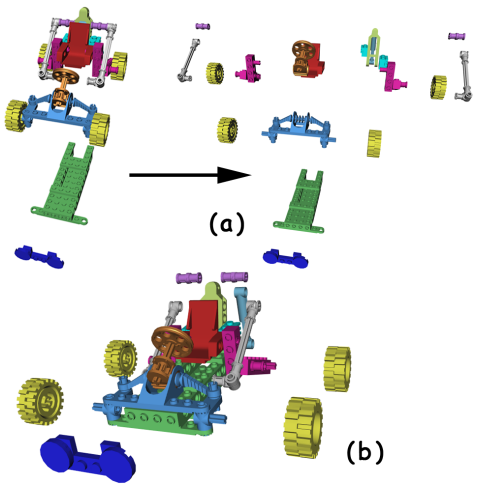


Figure 4: *Different relationships between parts results in different layouts of the explosion diagram. (a) The stacks of parts to the left and the right in the back of the car have been related to the seat. The wheels in the front of the car follow the blue steering gear. (b) The front wheels have been related to the base-plate of the car, as have both purple elements, which connect the wheels in the back to the car.*

collisions between exploding parts, the directions in which a part can be displaced are restricted to only those in which a part is not blocked by any other parts. This implies that the algorithm displaces parts which are unblocked in at least one direction, before it is able to explode parts which are blocked in all directions. Thus, by removing the exploded parts from the assembly, we gradually remove blocking constraints which allows us to explode previously blocked parts in a subsequent iteration of the algorithm. Since the algorithm gradually removes parts from the assembly, the set of directions for which a part is not blocked (and thus the set of potential explosion directions) depends on the set of previously removed parts. Consequently, the disassembly sequence directly influences the set of potential explosion directions.

Disassembly Sequence Previous approaches [Li et al. 2008; Agrawala et al. 2003] compute a sequence depending on how fast a part is able to escape the bounding box of the remaining parts in the assembly. However, since this approach does not comprise any information about the similarity between exploded parts, the resulting explosion layout does not ensure similar exploded views for similar assemblies. Consequently, we encode information about the similarity of the parts in the sequence. We remove similar parts in a row, starting with the smallest. If no similar part can be removed from the assembly, we choose the current smallest part. This strategy enables us to set up relationships which subsequently allow smaller parts to follow bigger ones during explosion. Take note that, by computing a larger amount of similar explosion layouts, our system is able to choose a representative exploded view out of a larger set of similarly exploding assemblies.

Figure 5 demonstrates the difference between previous approaches and our new strategy to find a disassembly sequence. A sequencing based on a bounding box intersection is demonstrated in Figure 5(a,b,c). The algorithm first removes part A before part B and part C will be exploded. By using this strategy, relationships between part A and part B and subsequently between part C and part B will be set up. The resulting explosion layout is illustrated in Figure 5(c). As can be seen, different explosion directions have been assigned to the similar parts B and C.

In contrast, our algorithm computes a sequence which is based on a comparison of the previously exploded part and all removable part in the remaining assembly. As demonstrated in Figure 5(d,e,f), our strategy will result in a sequence which supports similarly exploded views of similar assemblies. Both parts B and C have been displaced in the same direction and both parts have been related to the same part in the remaining assembly (part A).

Relationships Both strategies in Figure 5 set up relationships between the current part and the bigger one. However, since our sequence removes similar parts one after the other, the remaining assemblies are identical for similar parts, with the exception of the previously removed part (which is similar to the current one). Since almost identical conditions exist for similar parts, our algorithm is able to set up similar relationships for those parts and the parts in the remaining assembly.

In addition to the initial assignment of relationships between parts, we change the relationships for penetrating elements in a stack. For example, the black screws in Figure 3 have contact with the purple cylinder and the green gearbox. Since the green gearbox is the bigger item, the initial relation is set between a screw and the gearbox. However, this would result in an explosion diagram in which the screws follow the gearbox instead of the purple cylinder.

To handle such cases, we search for stacks of parts by searching for the elements which are located between the exploded part and the

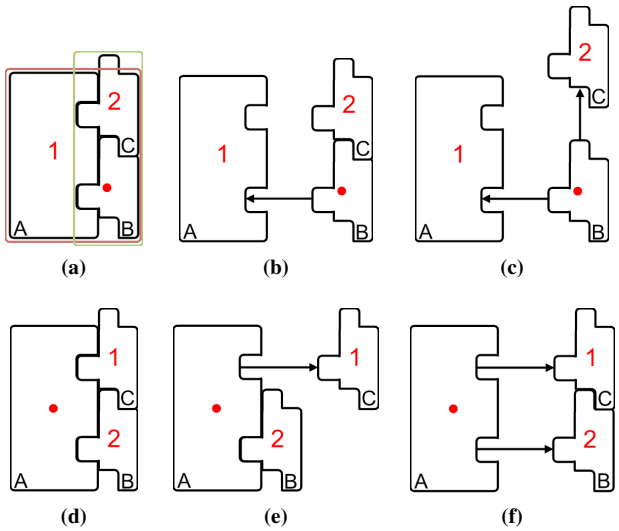


Figure 5: *Different disassembly sequences may result in different layouts. The sequence is labeled in red. The resulting explosion diagram is illustrated in the image on the right. (a,b,c) The sequence has been computed based on previous approaches which select parts depending on the distance a part has to be moved to escape the bounding of the remaining assembly. The bounding boxes of the remaining parts have been framed in red and green. (d) We compute the next element in the sequence based on a comparison with the previous one. (e) By removing similar parts in a row we ensure that the remaining assemblies contain the same elements, except for one part which is similar to the next one. (f) This strategy allows us to explode similar parts within similar conditions, which in turn results in more similar exploded views of similar subassemblies.*

one it is related to. If parts exist in-between and if these parts share an explosion direction with the currently removed part, the initial relationships are changed so that the exploded part is related to the closest part in the stack of parts in-between.

Explosion Directions Previous approaches compute the explosion direction of a part out of a set which contains only the six directions along the three main axes of the model [Li et al. 2008; Agrawala et al. 2003; Kalkofen et al. 2009b]. However, this approach is very limited (e.g. consider the differences in directions in the explosion diagram in Figure 3). Therefore, we compute a non-directional blocking graph, similar to the algorithm proposed by Wilson [Wilson 1992], by computing blocking information between all pairs of parts. For each exploded part, we determine the set of unblocked directions by removing all blocked directions from the set of exiting 3D directions. We represent all directions by a unit sphere and we remove blocked ones by cutting away the half sphere with a cutting plane which is perpendicular to the direction of a blocking part. By iteratively cutting the sphere, using all blocking information from parts in contact with it, the remaining patch of the sphere represent all unblocked directions for a part. Thus, we output the center of gravity from the remaining patch of the sphere.

3.2 Explosion Distance

If a subassembly appears multiple times in another subassembly, we introduce a hierarchy of subassemblies from which we choose representatives depending on an explosion style (see Section 5.3

for a discussion on hierarchical subassemblies). For example, the screws in Figure 3 form a cluster of screws which depends on the part they fasten. Each cluster consists of four screws which fasten a single cylinder. If a style is chosen, which explodes all screws in a single cluster, we have to compute a representative out of a higher level group of parts. Therefore, our system has to support an alignment of the distances of similar parts.

Since similar parts appear to be similarly large, we set the distance of displacement from the parent part to be proportional to the size of the exploded part. Nevertheless, since a linear mapping may easily result in very distant parts, we introduce a non-linear mapping using equation 1.

$$Distance = SizeOfPart.(1 - k.RelativeSize^2) \quad (1)$$

For parts which cannot be removed at all, we compute a distance where they can be moved until colliding with other parts. For example, the lower purple cylinder in Figure 3 cannot be removed before the black screws have been removed. However, the black screws will collide with the cylinder it fastens if we explode them into a single direction. Nevertheless, we can explode the screws a certain distance before they collide with the cylinder. Since this distance is sufficient to reveal the screws, we compute the maximal distance they can be exploded. We explode the screws a distance smaller than this maximal distance and we remove the screws from the assembly, so that we are able to subsequently explode the cylinder from the assembly.

We compute the maximal distance a globally locked part can be moved by rendering both parts - the one which is about to be removed and the one which blocks its mounting direction- into a texture. We position the camera at the vector along the explosion direction to point at the exploded part. In a vertex shader we use the current model-view transformation matrix to transform each vertex into camera space. The corresponding fragment shader finally renders the location of each fragment in camera coordinates into the textures. By calculating the difference between the texture values, we get a map of distances between the fragments of both parts. The maximal distance a part can be removed, before it collides with the blocking part, is finally represented by the smallest difference between the values in the texture.

4 Frequent Subassemblies

We determine sets of similar subassemblies by performing a frequent subgraph (FSG) search on a graph representation of the assembly. The implemented approach is based on the gSpan algorithm of Yan and Han [Yan and Han 2002], which uses depth-first-search (DFS) codes to differentiate two graphs. A DFS code describes the order in which the nodes of a subgraph have been visited. Two graphs are defined isomorphic if their DFS codes are equal and if their corresponding node labels (which represent the parts) match. Nodes of equal parts receive the same label and equally assembled objects generate equal DFS codes. By using DFS codes and node labels, the implemented FSG algorithm finds non-overlapping sets $S = \{G_1, \ldots, G_k\}$ of the largest subassemblies G contained in the graph.

4.1 Graph Representation of Assembly

To apply an FSG on the 3D model we have to transform it into an assembly graph Ag which contains all parts $P = \{p_1 \ldots p_n\}$, with n being the number of parts in the assembly. The nodes of the graph are the parts p_i (with $i = 1 \ldots n$) of the input model. If two parts are in contact, an undirected edge is created between the corresponding nodes.

We detect similar parts (and therefore the nodes which receive the same label) by exploiting the DESIRE shape descriptor of Vranic et al. [Vranic 2005]. The descriptor computes a feature vector for each part which we use to compare their shapes with. We consider two parts as being similar, if the l2-distance of their corresponding feature vectors falls below a certain threshold. Besides the set of node labels, the result of the part comparison is a list of disjointed sets of similar parts $P_s = \{p_i, \ldots, p_k\}$, for $i \neq k$, and $i, k < n$.

4.2 Frequent Subgraph Mining

Input to the algorithm is the whole graph A_g. All nodes of parts p for which no similar counterpart exist ($|P_s| = 1$), are removed from the graph. This ensures that only parts occurring more than once are considered. Then, for each set of similar parts P_s a set S_0 is created, containing $|P_s|$ number of groups G_0, each containing a single part $p \in P_s$. The sets S_0 define the starting parts for the FSG search. All of these sets have to be processed separately by the mining algorithm, to find all FSGs of the assembly. A recursive FSG mining procedure is applied on each of the sets S_0 and iterates through all groups G_i of its input set S_i. The algorithm now tries to grow the groups G_i in the following way.

In each iteration, a reference group G_r is chosen from S_i. For G_r the set of neighbors N_r of the part last added to the group, are retrieved. If all neighbors of this part have already been visited, the ones of the previous part are chosen. If no unvisited neighbor is found, then all neighbors have been visited and the group G_r cannot be extended further. For each G_i, those neighbors similar to the ones in N_r are determined. Neighbors are similar to each other if their labels and number of contact parts to the corresponding group G_i are equal. Furthermore, the DFS codes and labels of those contacts have to be equal. This similarity measure ensures that the structures of the found groups are equal.

If at least two similar neighbors n_i and n_j are found for different groups G_i and G_j a new set S_n is created containing the groups $G_{n1} = G_i \cup n_i$ and $G_{n2} = G_j \cup n_j$. This is done for all groups having similar neighbors. Note, that for each set of similar neighbors a new set of groups is created and these groups differ only by one part from the groups of S_i. Hence, by recursively calling the mining procedure on the new sets, a DFS is performed, growing these groups further.

After all neighbors have been processed, each group which was extended by a similar neighbor, is removed from the set S_i because these groups have become subgroups of larger groups. If $|S_i| = 0$ all groups were extended and the set is deleted. If $|S_i| = 1$ the set is also deleted, but the group is processed further, because it may still contain smaller similar subgraphs. Therefore, the mining algorithm is applied again to eventually extract these graphs.

The FSG mining returns with the sets S_o of largest similar groups G_o. Overlapping output sets are resolved by keeping only one of the overlapping sets S_o and applying the FSG again to the set of $A_g \setminus S_o$. This operation is repeated for all results, until the output sets S_o do not overlap anymore. We keep the one overlapping set which contains the groups holding the most number of parts. If this measure is ambiguous, the set having the most groups is preferred. If this is still ambiguous the one containing the largest part is chosen.

4.3 Group-based Layout

Our system calculates similar subassemblies independent from the initial layout of the explosion diagram. However, even though our sequence generator specifically supports similar exploded views of

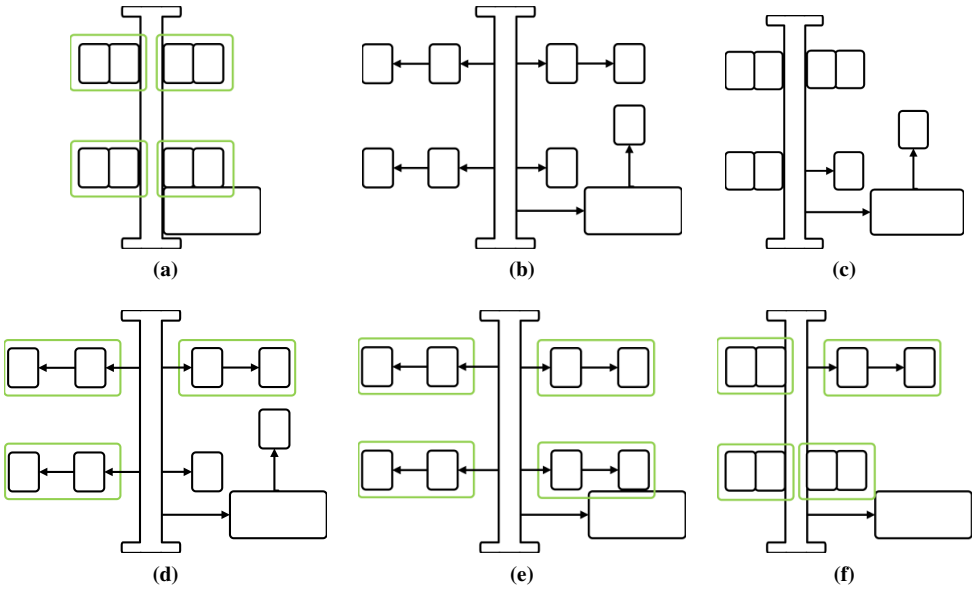


Figure 6: *Explosion Layouts containing group information. (a) Groups have been created independently from the explosion layout. (b) Therefore, the explosion layout does not take information about similar subassemblies into account. This may generate different exploded views of similar subassemblies. (c) If we select a representative from a set of similar subassemblies which do not explode the same way, the explosion does not demonstrate all other subassemblies. (d) By recalculating group information from the layout, the number of similar groups is reduced which results in more exploded views. (e) Therefore, we modify the initial layout so that similar subassemblies explode in a similar way. (f) This strategy allows us to choose a representative from a larger set of subassemblies which in turn reduces the amount of required exploded views to demonstrate the assembly.*

similar subassemblies, if the neighborhood of both differ, the exploded views may be different. For example, the model in Figure 6(a) consists of one set of four similar subassemblies (marked by the green rectangle). Each of them contains two parts. Figure 6(b) shows its explosion diagram in which each single part has been displaced. As can be seen from the initial layout, the exploded view of the subassembly in the lower right corner is different from the other. If we choose this exploded view as the representative of its set of similar subassemblies, the resulting compact explosion diagram lacks a presentation of the other subassemblies of this set (Figure 6(c)).

To prevent representatives which explode differently to other similar subassemblies, we can adjust the sets of similar subassemblies in a way that only similarly exploding subassemblies will be grouped together. Therefore we use the layout information to modify the identification of similar subassemblies. Only those parts of the assembly are candidates for extending a group which would set up a relationship to another part in the subassembly. Figure 6(d) shows the result of this restriction. This strategy finds a set of only three instead of the previously identified four similar subassemblies (marked in green). Consequently, less subassemblies will be presented assembled which results in a layout which is not as compact as in the previous case.

In order to create a more compact explosion layout, without risking to choose a representative which does not demonstrate the composition of other similar subassemblies, we modify the layout of the explosion diagram instead of the information about the similarity of subassemblies. As illustrated in Figure 6(e) we aim to modify the layout to prevent relationships with parts outside the subassembly. We allow only one relationship between a part in the subassembly and the remaining 3D model.

Note, this is similar to the approach of Li et al. [Li2008] who explode a manually defined group of parts as if it was a single element in the assembly. However, we use a different approach to handle interlocking groups. Rather than splitting a subassembly, we ignore blocking parts. This allows us to keep subassemblies connected. Note, this could be at the cost of explosion diagrams which are not completely free from collisions. Nevertheless, we believe that preventing such collisions is less important for the final compact explosion layout than a larger amount of explosions or a representative which does not demonstrate the composition of its associated subassemblies. In the case of a compact explosion diagram, it is more important to select a representative from a rather large set of similar subassemblies, which additionally all explode in a similar way.

Thus, we compute an explosion diagram which ensures similar explosion layouts of similar subassemblies as explained in section 3. However, for each part p_i we determine if it is a member of a subassembly G_i which occurs multiple times in the model. If the algorithm is about to explode a part p_i which is a member of G_i, we choose a representative part p_r out of G_i which we explode instead of p_i . We define p_r as the biggest part in the subassembly G_i which has at least one face in contact with at least one part of the remaining assembly, not considering other parts of the subassembly. In addition, the representative part p_r has to be removable in at least one direction without considering blocking constraints of parts of the same subassembly.

Even though p_r influences the explosion direction of the entire subassembly, we may not set the relationship between p_r and a part out of the remaining assembly. Since we are only able to explode each part once and since we want to further continue to explode all frequent subassemblies in the same way, we have to choose the same part in each subassembly to set up the relation to the remaining assembly. Moreover, since we want to explode subassemblies using

the guidelines presented in section 3, we want to explode the small parts before the bigger ones. Therefore we choose the biggest part in the assembly as the main part of the assembly and we relate it to the biggest part in the remaining assembly which the subassembly has contact with.

If frequent subassemblies exist in an exploded subassembly we cannot simply search for the bigger part in the main subassembly, because we also want to create a similar exploded view of all frequent subassemblies, even if they appear cascaded. Instead, we first compute a hierarchy of subassemblies (see section 5.3 for details) before we choose the biggest part from only the highest level of the hierarchy. The highest level ensures that no other part is similar to the chosen one and consequently no conflicting explosion layout can result. Note once again, by removing entire subassemblies in an unblocked direction of a single representative member we ignore collisions between parts during explosion. Even though this may result in physically incorrect sequences to disassemble the object, we are able to explode subassemblies independent of the overall model, which in turn enables to calculate a single explosion layout for all similar subassemblies.

5 Selecting Representatives

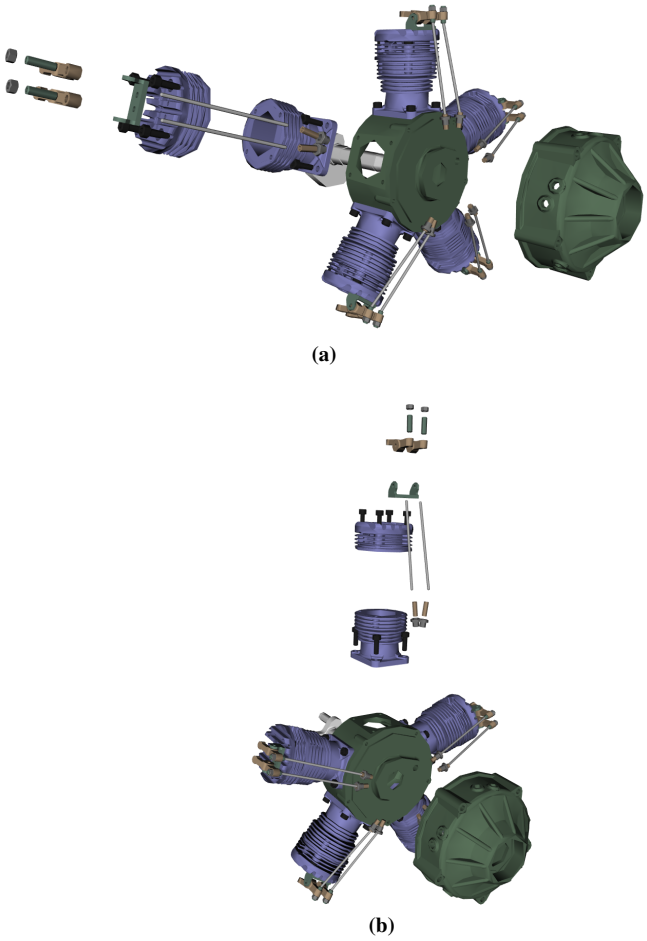
After identifying frequent subassemblies and after computing an initial explosion layout, a compact representation is created by displacing only one representative group out of a set of similar groups. We select a representative subassembly by calculating its quality as the weighted sum of a set of different measurements (section 5.1). Since the combination of representatives may influence the measurements of a single subassembly, we optimize the selection based on the idea of threshold accepting [Dueck and Scheuer 1990] (section 5.2). In the following, we will first describe our algorithm to select a representative subassembly out of a set of similar subassemblies, before we present our approach to combine representatives to the final compact explosion diagram

5.1 Quality Measurements

We define the quality of a group of parts as a combination of several measurements. Therefore, for each group we render its local explosion (which displaces only the parts of the group and parts which block the group) and we compute the following values:

- *Size of footprint of the exploded group.* The footprint f describes its size in screen space. The footprint is used to estimate the overall visibility of a part, given its position and orientation.
- *Visibility of parts of the exploded group.* The visibility v is a relative measure for the general visibility of the parts.
- *Part directions relative to current camera viewpoint.* Assuming that explosions, which are similar to the viewing direction, are more difficult to read than those which explode more perpendicular to the viewing direction, we compute the dot product a between the viewing vector and the explosion direction for each part. The average value of all values a is used as the value for a group.
- *Size of footprint of all other similar groups without any displacements.* This value describes how well other similar and unexploded groups will be visible, if the tested one will be chosen as the representative. Therefore, footprint f_r of the unexploded groups is determined.

The final score Q_r of an exploded view of a subassembly consists of the weighted sum of these four measurements (see equation 2).



(a) **(b)**

Figure 7: *Differently weighted scoring parameter. (a) Highly weighted visibility of representative groups. The algorithm favors exploded views which cover larger amounts of screen space. (b) By weighting the dot product between the viewing direction and the explosion direction higher, the algorithm selects representatives which explode more perpendicular to the viewing direction.*

The weights (f_c, v_c, a_c, f_{rc}) indicate the importance of each single parameter to describe the quality of the group. Since an evaluation of ideal weights is left for future work, our system allows to interactively modifying them.

$$Q_r = f.f_c + v.v_c + (1 - a).a_c + f_r.f_{rc} \qquad (2)$$

Figure 7 shows the results of two differently weighted scorings. Since we compute the score of a group locally, visual overlaps of representatives (such as in Figure 8(a)) or interdependent explosion, which change the score of a group are not considered at this point of our computations. The following section describes how we optimize such combinations.

5.2 Combining Representatives

To avoid the interference of representatives with each other, we apply the search for an optimal combination of exploded groups using the idea of threshold accepting [Dueck and Scheuer 1990], a heuristic optimization strategy. In each optimization step, the quality of a different combination of exploded representative groups is evaluated by adding their scores.

The initial layout consists of exploded representatives with the highest local scores. Therefore, if the sum of their local scores is equal to the global score, the local representatives are global representatives too. Consequently, we do not search further for a better combination. However, if the global score is less than the sum of local scores, we change the initial layout by a single representative group and recompute the global score of the modified layout. If the score of the changed layout is higher than the current best finding, this new one is used as the current best combination of representatives. Therefore, if the new score is equal or less than the current best score, we do not consider the current combination to be displayed. However, even if the current score is less than the best one, we compute the next tested layout based on the current one, if its difference to the best score is less than a threshold value. Otherwise, we modify the layout which the current layout was computed from. While the algorithm progresses, the threshold value decreases, which gradually allows better layouts to be the starting point for further changes.

Figure 8(b) shows the results of optimizing the locally scored compact explosion diagrams presented in Figure 8(a). Since representatives in Figure 8(a) overlap each other, a different group has been selected in the optimized compact explosion diagram in Figure 8(b). Figure 8(c) shows a compact explosion diagram where the visibility of unexploded parts was weighted higher.

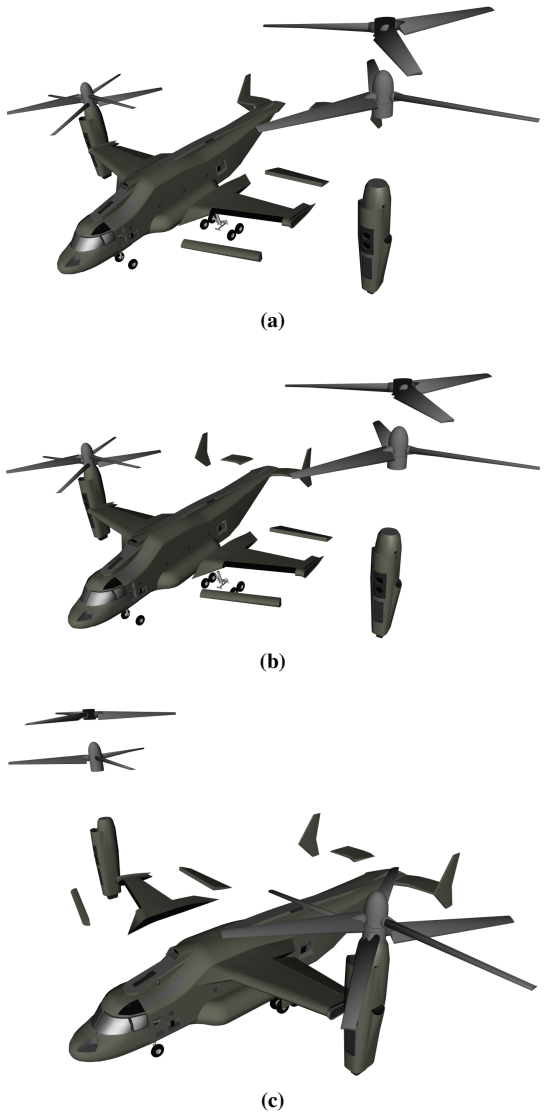
5.3 Hierarchical Subassemblies

After applying the FSG search to the graph A_g of the whole assembly, a list of sets which contain the largest available non-overlapping subassemblies has been discovered. However, the selected subassemblies may even contain other frequent subassemblies. If we also identify these subassemblies we are able to select a representative in multiple levels of the hierarchy, which in turn allows us to further reduce the number of displaced parts in a representative exploded view. To find frequent subassemblies within a previously determined subassembly, we apply the FSG algorithm recursively until no subassembly can be determined anymore. When performing the FSG search on a set S of groups G, each group G is considered to be a separate graph to be mined for subassemblies. This means that a subsequent FSG search does not exceed the limits of the groups they are applied to.

5.4 Similar Neighborhoods

By recursively applying the FSG search algorithm to a subassembly we receive a hierarchy of frequent subassemblies. The groups of the detected sets and subsets are similar to each other, because their graph representations are isomorphic. However, groups of the same set may have different neighborhood relations to the group they are contained in. The reason for this is that the FSG mining algorithm removes all parts from the input graph, which do not have similar counterparts. Basically, this removes the neighborhood between a subgroup and the group it is contained in. By recovering this information, we are able to refine the hierarchy. This allows us to choose better representatives from a set, because similar groups are then also distinguishable by their neighborhoods. Therefore, we define that similar groups G_l not only have to be similar in terms of graph isomorphism, but also the neighborhood to the groups G_h they are contained in has to be similar. We implemented the following algorithm, which searches for similar neighbors of groups of a set.

For each neighbor of a group the set of adjacent groups E_n is determined. Sets E_n of similar neighbors in different groups G_h are merged into the set E_s. Then, simple set operations are performed on the sets E_s to retrieve the common neighborhood for similar groups. For a representative E_r from the sets of E_s, the following operations are performed in combination with each other E_s. First, the intersection $E_c = E_r \cap E_s$ is created. If $|E_c| = |E_r|$, all groups share the same neighbor and the algorithm continues. Otherwise, the groups of E_r share different neighbors. These groups are eliminated from E_r ($E_r = E_r \setminus E_c$). The algorithm continues until either all E_s have been considered, or $|E_r| = 0$. Those groups left in E_r have similar neighborhoods. The algorithm finally terminates when all sets of E_s have been considered as representative set E_r.



(a) **(b)** **(c)**

Figure 8: *Optimized Selection of Representative. (a) A local scoring using high weighting factors for visibility information of the exploded parts may result in an overlap of exploded parts. Notice the hidden wing at the back of the plane (b) We compute an optimized layout by selecting new combinations which we accept based on a dynamic threshold. (c) The systems calculates an optimized compact explosion diagram using higher impact of visibility of unexploded parts.*

5.5 Selection in Hierarchical Groups

If a hierarchy of groups exists, we allow to select representative exploded views using three different strategies. We allow either to choose the representative parts from a single subassembly (Figure 9(a), Figure 9(b)), or to select representative parts independently in different subassemblies of the same set (Figure 9(c)). If we chose to restrict the explosions to a single hierarchy, we have to decide if we want to explode the entire subassembly (Figure 9(a)) or only a single representative in each level of the hierarchy (Figure 9(b)). Figure 9 shows an example for each given situation. Since it is an open question which strategy results in the perceptually best results, our system allows selecting a strategy at runtime. We leave a perceptive evaluation of the comprehensibility of each strategy for future work.

6 Conclusion

Explosion diagrams provide a powerful technique to visually communicate the composition of 3D objects. However, if complex models have to be presented, the explosion diagram may suffer from clutter which is caused by the excess of displaced parts. To reduce the complexity in explosion diagrams, illustrators often chose to displace only a subset of parts. In this paper we have presented a system which is able to automatically mimic this technique. We have presented an algorithm to find similar subassemblies in a 3D object. We discuss several strategies to ensure that a representative exploded view demonstrates the unexploded remaining subassemblies. In addition, we have presented a new quality measure for exploded views of subassemblies. We have demonstrated the impact to the resulting explosion layout of our quality parameters. Additionally, we introduced a strategy to calculate optimized combinations of representative exploded views, which we discussed in different examples, which demonstrate its performance using differently weighted quality parameters. We have presented a new algorithm to compute a disassembly sequence which allows us to compute an explosion layout in which similar subassemblies have been exploded in a similar way.

Our work focuses on assemblies which consist of recurring subassemblies. However, even if no such subassemblies exist, the calculated explosion diagram does not differ from any other exploded view which displaces all of the parts of the input model.

In a future system we will extend the set of quality parameters e.g. a contrast information between exploded parts and their background information. We will furthermore evaluate the impact of the different quality parameters on the comprehension of the demonstrated subassemblies. In addition, we will extend the information about groups of parts to also consider the cluster of recurring subassemblies which do not have direct contact with each other.

Acknowledgements

This work was sponsored in part by the Austrian Science Fund FWF under contract W1209 and by the Christian Doppler Laboratory for Handheld Augmented Reality. We would like to thank D.V. Vranic and T. Schreck for providing us the DESIRE shape descriptor implementation.

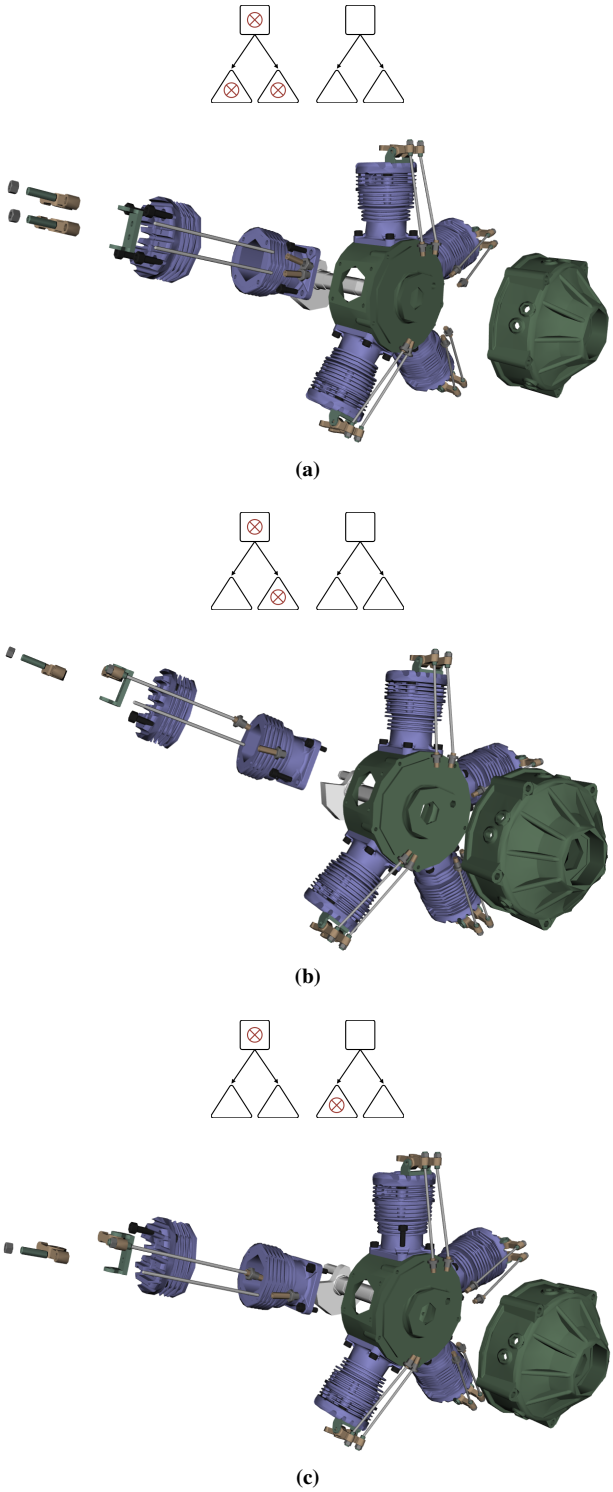


Figure 9: *Selection Strategies in Hierarchical Groups. (a) All parts in a subassembly have been exploded. (b) Representatives have been selected in each level of the hierarchy. (c) Representatives have been selected in different subassemblies.*

References

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, 828–837.

BRUCKNER, S., AND GRÖLLER, M. E. 2006. Exploded views for volume data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5, 1077–1084.

DUECK, G., AND SCHEUER, T. 1990. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics 90*, 1, 161–175.

KALKOFEN, D., MENDEZ, E., AND SCHMALSTIEG, D. 2009. Comprehensible visualization for augmented reality. *IEEE Transactions on Visualization and Computer Graphics 15*, 2, 193–204.

KALKOFEN, D., TATZGERN, M., AND SCHMALSTIEG, D. 2009. Explosion diagrams in augmented reality. *IEEE Virtual Reality Conference 0*, 71–78.

LI, W., AGRAWALA, M., AND SALESIN, D. 2004. Interactive image-based exploded view diagrams. In *GI '04: Proceedings of Graphics Interface*, 203–212.

LI, W., RITTER, L., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2007. Interactive cutaway illustrations of complex 3d models. In *Proceedings of ACM SIGGRAPH 2007*, ACM, New York, NY, USA, 31–39.

LI, W., AGRAWALA, M., CURLESS, B., AND SALESIN, D. 2008. Automated generation of interactive 3d exploded view diagrams. *ACM Transactions on Graphics 27*, 3, 1–7.

MIJKSENAAR, P., AND WESTENDORP, P. 1999. *Open Here. The Art of Instructional Design*. Thames & Hudson.

NIEDERAUER, C., HOUSTON, M., AGRAWALA, M., AND HUMPHREYS, G. 2003. Non-invasive interactive visualization of dynamic architectural environments. In *Proceedings of the Symposium on Interactive 3D Graphics*, 55–58.

RAAB, A., AND RÜGER, M. 1996. 3d-zoom: Interactive visualisation of structures and relations in complex graphics. In *3D Image Analysis and Synthesis*, infix-Verlag, H.-P. S. B. Girod, H. Niemann, Ed., 125–132.

RIST, T., KRÜGER, A., SCHNEIDER, G., AND ZIMMERMANN, D. 1994. AWI: A workbench for semi-automated illustration design. In *Advanced Visual Interfaces*, ACM Press, New York, NY, USA, 59–68.

RUIZ, M., VIOLA, I., BOADA, I., BRUCKNER, S., FEIXAS, M., AND SBERT, M. 2008. Similarity-based exploded views. In *Proceedings of Smart Graphics 2008*, 154–165.

SONNET, H., CARPENDALE, S., AND STROTHOTTE, T. 2004. Integrating expanding annotations with a 3d explosion probe. In *Advanced Visual Interfaces*, ACM Press, New York, NY, USA, 63–70.

VRANIC, D. V. 2005. Desire: A composite 3d-shape descriptor. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 962–965.

WILSON, R. H. 1992. *On Geometric Assembly Planning*. PhD thesis, Stanford University, Stanford, California.

YAN, X., AND HAN, J. 2002. gSpan: Graph-based substructure pattern mining. In *Proceedings of the IEEE International Conference on Data Mining*, IEEE Computer Society, Washington, DC, USA, 721.