Explosion Diagrams in Augmented Reality

Denis Kalkofen*
Graz University of Technology
Institute for Computer Graphics and Vision

Markus Tatzgern†
Graz University of Technology
Institute for Computer Graphics and Vision

Dieter Schmalstieg‡
Graz University of Technology
Institute for Computer Graphics and Vision

ABSTRACT

This article introduces explosion diagrams to Augmented Reality (AR) applications. It presents algorithms to seamlessly integrate an object's explosion diagram into a real world environment, including the AR rendering of relocated objects textured with live video and the restoration of visual information which are hidden behind relocated objects. It demonstrates several types of visualizations for convincing AR explosion diagrams and it discusses visualizations of exploded parts as well as visual links conveying their relocation direction. Furthermore, we show the integration of our rendering and visualization techniques in an AR framework, which is able to automatically compute a diagram's layout and an animation of its corresponding explosion.

Keywords: Object overlay and spatial layout techniques, real-time rendering, mediated and diminished reality.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities E.1 [Data Structures]: Graphs and Networks I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

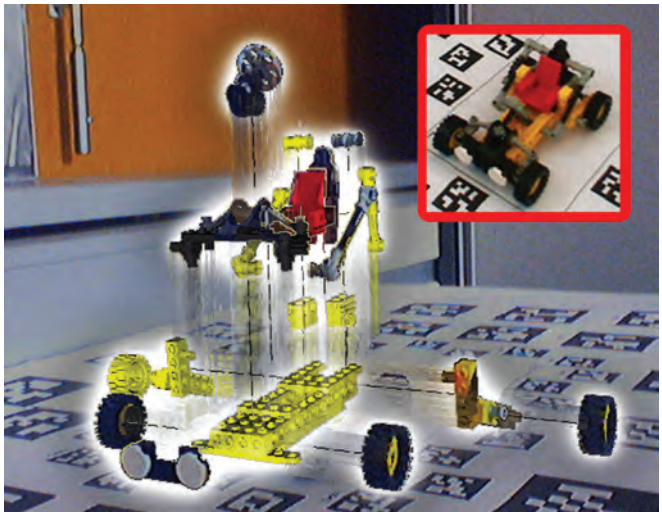


Figure 1: Explosion Diagram in Augmented Reality. Real world information is displaced using *synchronized dual phantom rendering*. Exploded parts are visually unified using either pure virtual or real world combined with restored real information. A halo outline discriminates the parts from background information and their connection lines are embedded in a motion blur effect resulting from their explosion.

1 INTRODUCTION

Augmented Reality (AR) often draws its power from its ability to seemingly suspend physical laws of light propagation in real world environments. For example, AR can reveal hidden structures by so called *x-ray visualization*, i. e., by rendering virtual representations of hidden objects registered in three-dimensional real space. However, overriding real world imagery with virtual objects can hide important information in the original image which may lead to ambiguous depth perception. For example, the simple overlay of a hidden object used in Figure 2 makes it difficult to estimate the distance between the car's engine and its front lid.

1.1 X-ray visualizations and their limitations

To correctly communicate spatial arrangements, x-ray visualizations have to take into account the information which is about to be occluded by the overlaid virtual objects. Recent work on visualizations for AR [12]. shows how the presentation of contextual information is able to support the comprehension of x-ray images. Two types of visualization techniques have been proposed in AR literature so far: *cutaways* [20] are artificial cavities in an occluder, while *ghostings* [12] use sparse representations of the occluding structures to allow to see inside or behind.

Both techniques preserve depth cues by retaining some portion of the occluding structure as contextual information, and only work efficiently for suitable occluders. For example, cutaway visualizations require that the occluding object is large enough in screen space to contain the virtual cavity. Ghostings require the presence of appropriate surface features, which can be extracted to yield the desired sparse representation. Too few or too many features are both detrimental as they lead to either empty or cluttered displays. Introducing artificial surface features, for example based on textures such as suggested by [13], is not applicable in AR where real world object textures are already present and must be preserved if possible.

Even more problematic is the fact that both ghosting and cutaway renderings remove potentially important information from the occluding objects. Kalkofen et. al. [11] show how important context information can be preserved in a ghost visualization. However, in densely cluttered, complex scenes, where several important focus and context objects must be shown, attempting to visualize all relevant objects in a very small space, satisfactory results may still not be yield.

1.2 Advantages and disadvantages of explosion diagrams

As a remedy, we suggest to use explosion diagrams in AR (Figure 1). Explosion diagrams are traditionally used in technical illustrations to present the assembly of an object. Unlike cutaway or ghost visualizations, they avoid the trade-off between retaining occluders and uncovering hidden structures. Instead, objects are displaced and presented using an arrangement that makes it possible to mentally reassemble the object. Explosion diagrams therefore maximize the amount of relevant visual information, while the communication of spatial arrangements is still well supported.

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

IEEE Virtual Reality 2009
14-18 March, Lafayette, Louisiana, USA
978-1-4244-3812-9/09/$25.00 ©2009 IEEE

Figure 2: X-ray vision. Simple overlay of hidden structure

The key element of traditional explosion diagrams is their layout, showing the exploded parts relative to each other. This spatial arrangement encodes the relationships of the parts, so that the observer can mentally reassemble the exploded object. However, explosion diagrams in AR must respect the relationship of virtual and real components of the scene, and therefore require special rendering and visualization techniques. Specifically, we must deal with the problem of convincingly relocating real world objects and fill the resulting empty spaces with equally convincing virtual information.

Furthermore, traditional illustrations usually present the explosion diagram in front of a uniformly colored background. In contrast, AR displays must take into account the real world (video) background and ensure that all components of the explosion diagram (the relocated objects and the visual links between parts that convey their assembly sequence) stand out from the background in a clear and unambiguous way. As an example, consider how difficult it is to see a single colored dashed line Figure 5(e).

1.3 Contribution

In this paper, we address the task of integrating explosion diagrams in an AR environment. We present algorithms to compose an image from exploded real world information, non-exploded real world information and virtual objects (section 3). We demonstrate the restoration of missing hidden information in cases where the AR visualization suffers from a deficiency of information after relocating real world imagery. To support the comprehension of an AR explosion diagram, we discuss different types of visualization techniques of the parts of a diagram as well as how to visually link them (section 4). We integrate our rendering and visualization techniques in an AR framework, which is able to automatically compute task dependent layout and animation of the explosion diagrams (section 5).

2 Related Work

Explosion diagrams can be found in may different media, ranging from illustrations in books to interactive scientific visualizations of volumetric data. Most of them are used to support the understanding of an exploded object. For example, Ritter et al. [18] present a 3D puzzle, which helps the user learning spatial relationships of the parts of an object. They create an 'in-place' explosion by scaling down the parts of an object, a similar technique as presented by Raab [17].

In psychological studies the effectiveness of design parameters on the comprehension of an explosion's layout was carried out for the case of still renderings presenting an assembly plan [9]. Given a 3d CAD model and a set of semantic information about the geometry (like its included groups of parts), applications are able to follow the rules to automatically create static images of comprehensible explosion diagrams [1].

The layout in real time explosion diagrams is usually closely coupled with an interaction technique. For example, Sonnet et al. [21] utilize a visual distortion technique bound to a 3D frustum which is defined by the image plane and a movable 3D probe. Interacting with the probe enables the user to explode all parts out of its line of sight. Brucker et al [5] present interactive tools to divide volumetric data in a set of parts, which afterwards explode apart by applying a set of forces to them. McGuffin et al. [16] present a toolbox of interactive techniques, also applicable on volumetric data, but instead of linearly moving the parts, they deform the underlying data.

Besides explosion diagrams from 3D CAD data or volumetric data sets, Li et al. [15] have presented tools to create interactive explosions in image space. In a very recent publication [14], he extended his system to create interactive explosion diagrams in 3D where he uses a system similar to [1].

While already a number of systems exist to create interactive explosion diagrams, none of them had to deal with real world information in their presentation space. Moreover, none of the presented systems is able to compute a task-dependent explosion layout fully automatic, without any additional information than that derived from the loaded CAD data.

3 Rendering Explosion Diagrams

Explosion diagrams in AR consist of real, virtual and relocated real information. To correctly compose an image out of all three types of information, the rendering algorithm has to achieve three requirements. Firstly, it must be able to convincingly relocate real world structures. Therefore, visual information has to be transferred from its original to the target location after the explosion was applied. Secondly, new imagery has to be generated to fill the original locations. Thirdly, the rendering algorithm has to correctly resolve occlusions between all used data. In this section, we present algorithms to relocate real world information, which fulfill these three requirements and we also discuss the advantages and disadvantages of these approaches.

3.1 Video-Textured Phantoms

Convincing AR renderings must resolve occlusions between virtual and real world objects. To find out which fragments are visible, a common approach is *phantom rendering* [4] which uses the depth values of real world objects which are computed by adding the virtual counterpart, the phantom object, of real objects to the scene description. Phantom objects are rendered invisibly, only to the z-buffer, which enables the application to subsequently resolve occlusion among real and virtual objects using the z-buffer, assuming that the phantom objects are properly registered with their real world counterparts. While phantoms themselves are rendered fully transparent, the real world information from the video background is kept where a phantom occludes all fragments from virtual objects.

Since simple phantom rendering does not take into account the relocation of objects, it is not suitable for our goal of combining real, virtual and relocated real objects. Specifically, information from the video background in the color buffer must be transferred to its new location when the phantom of a relocated object is rendered. Thus, we extend the original idea of rendering transparent counterparts of real world objects to video-textured phantoms (Figure 3(a), Figure 3(b)).

To texture a phantom object with video information, we calculate the u,v coordinates for each fragment as if the video background was applied using projective texture mapping from the cam-

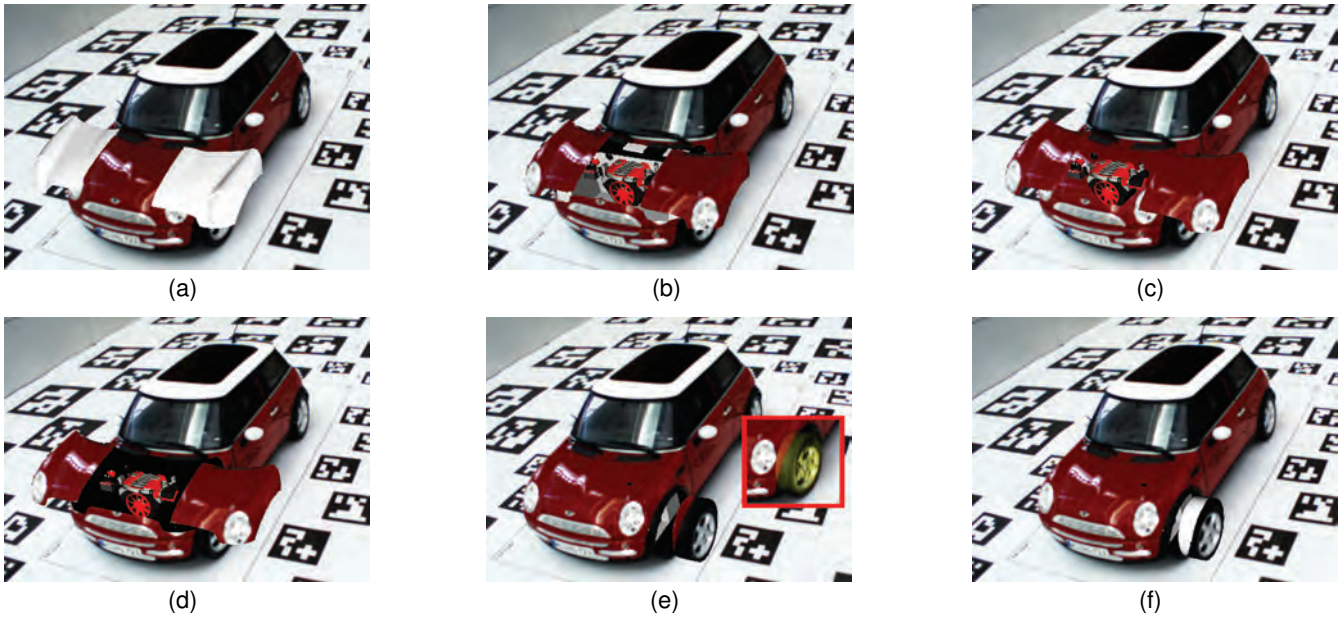


Figure 3: Distributing real world information. Exploded virtual phantom object (a) Transferred real world information to the phantom object (b) Incomplete virtual scene and phantom rendering (c) Dual phantom rendering is used to remove void information (d) Phantoms with overlapping 2D footprints using the same video information (e) Synchronized dual phantom rendering to control the usage of video information (f)

era's point of view. This is implemented by simply multiplying each vertex of a phantom with the combined model-view-projection (MVP) matrix representing the model (the phantom) before explosion transformations are applied.

Using a vertex shader, this can be achieved in two ways. We can either render the phantoms in their real world location and pass the transformation to the new location to the shader, or we compute the MVP matrix for each phantom beforehand and use it in the shader while the phantom is rendered in its new location. Since the second approach fits better into a scene graph framework, leveraging its ability to cascade transformations, we choose it in our implementation. Consequently, we pass the matrix to transform a vertex from object to world space before the explosion's transformation is applied. The necessary interpolation of the calculated u,v coordinates is performed by passing the calculated values from the vertex shader to the pixel shader. Note that interpolation of u.v coordinates rather than color values is necessary to avoid artifacts.

Since all objects are rendered only once and no shading is used, the computation of pixel colors only consists of a calculation of texture coordinates and a lookup of the current video feed per fragment. Therefore, rendering of video-textured phantoms has negligible overhead compared to simple phantom rendering and also works if no relocation is applied.

3.2 Dual Phantom Rendering

With video-textured phantoms we can relocate real world objects to another location in the image, thereby revealing the virtual objects behind the relocated objects. This assumes that the complete area of the relocated object is covered with virtual objects, which overrides the part of the image originally covered by the relocated object. However, frequently only a part of the uncovered area is occupied by a virtual object. Without special measures, the remaining area will still show the original video image (Figure 3(c)). We must therefore extend the algorithm to invalidate any relocated real world information in its original location, to be able to either create a cut-out or to supplement incomplete hidden information (Figure 3(d)).

To identify invalid pixels, we add a second render pass in which we project all fragments of a phantom onto their original real world location. This generates a 2D mask, consisting of only those pixels which will occur twice in a simple video-textured phantom rendering. This mask can then be used to either remove redundant real world information, resulting in e.g. a black background where no information is available, or the mask can be used to supplement incomplete hidden structures (section 3.4). The algorithm, called *dual phantom rendering*, can be described as follows:

1. *Enable and initialize framebuffer-object*
   a) Enable rendering to target 1 (T1)
   b) Clear depth buffer and render target (T1 is cleared with 100% transparent pixel)
2. *Render all video-textured phantoms (as described in section 3.1) to T1*
3. *Render all virtual objects to T1*
4. *Switch rendering to target 2 (T2)*
5. *Render all phantoms in its original location to T2*
6. *Disable render-to-framebuffer-object and switch back to on-screen rendering*
7. *Fill the color-buffer with the current video feed*
8. *Cut out invalid real world information using T2*
9. *Superimpose T1*

Note that in step 5 of our algorithm we are only interested in a binary 2D mask. This allows us to disable shading, thereby accelerating the rendering process. Furthermore, in cases where only a simple cut-out (and no restoration) of invalid real world information is desired, steps 7 and 8 can be combined by filling the color buffer depending on the 2D mask of the invalid video pixel (T2).

The algorithm as outlined only marks those fragments as invalid that will be visible in the final composition. This is controlled by the

values in the depth buffer after virtual objects- and video-textured phantoms are rendered (after step 3). Not touching the depth buffer before rendering the phantom objects (step 5) allows us to reject all fragments which are hidden by either virtual or relocated real world information. This is an appropriate approach for those cases where a simple cut out of invalid information is desired. However, if the restoration of hidden information is requested, a 2D mask representing the entire phantom object produces better results, because it presents the 2D footprint of the entire object and not only those of its visible portion. Such a 2D mask can be computed by clearing the depth buffer before phantom rendering is initiated (before step 5).

Even though dual phantom rendering can be accelerated in many cases, it still incurs a considerable performance overhead because of the required second rendering pass. Therefore, this approach is only recommended if required by the AR application.

3.3 Synchronized Dual Phantom Rendering

Labeling transferred video information in those places where a part of an explosion has been originally located, enables us to remove redundant information. However, in those cases where phantoms overlap in screen space, we will still transfer the same real world information to more than one object (Figure 3(e)). To completely avoid duplicate usage of real world information, we have to further restrict the transfer of information to only those fragments of the phantom that are actually visible in its original location (Figure 3(f)).

Therefore, instead of directly texturing a relocated phantom, we will first render the phantom at its original location, as in the previous approach. However, instead of simply marking the information as invalid, it is labeled with the phantom's object ID. By using regular OpenGL depth tests we obtain an ID buffer of only visible fragments. This ID buffer allows us to restrict the transfer of video information to only those fragments which have been identified as visible in their original location. The algorithm to synchronize the transfer of real world information can be outlined as following:

1. *Enable and initialize FBO*
   a) Enable rendering to target 1 (T1 = ID-Buffer)
   b) Clear depth buffer and render target

2. *Render IDs of all phantoms in its original location to ID-Buffer (T1)*

3. *Disable FBO / Switch back to on-screen rendering / Clear depth buffer*

4. *Fill color-buffer with the current video feed*

5. *Cut out invalid real world information using the ID-Buffer as 2D mask (phantom ids >0)*

6. *Render all video-textured phantoms. Use ID-Buffer (T1) to control the usage of video information*

7. *Render all virtual objects*

While Synchronized Dual Phantom Rendering requires, next to a second render pass an ID buffer, we favor this approach over an unsynchronized Dual Phantom Rendering only in scenarios where the phantoms may overlap in screen space. Even though most of the real world scenarios consist of a set of objects which overlap in 2D, some applications may focus on only a subset allowing the use of the simpler unsynchronized dual phantom rendering.

3.4 Restoration

Since the video feed of an AR system delivers only information about visible real world objects, their rearrangement may introduce spots without any available information. Virtual objects are used to fill out these empty places, but often the virtual model fails to completely cover this area (Figure 4).

We therefore utilize a restoration technique to fill in empty areas resulting from relocating real world objects. In our current implementation we identify the background information on the border of a mask, resulting from a relocation of parts of an object. The empty area is filled using the mean value of the all identified real world background information (Figure 5).

This technique was chosen because it is simple and fast, and leads to acceptable results for the applications we are currently considering. However, more advances techniques such as inpainting [2] exist, but are left as future work.

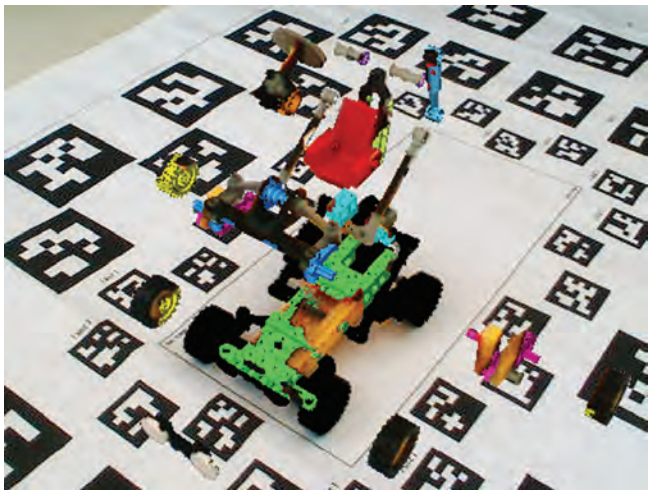


Figure 4: Bad example of an explosion diagram in AR. No further shading of the transferred real world information is used. Notice the clutter of real and virtual information.

4 Visualization

While conventional illustrations can arrange all visual elements on a uniform background, AR visualizations must deal with cluttered real world images. A relocated object will therefore not only be moved out of its natural context, but will be transported in a new, potentially even worse context. The new context is usually not related to the exploded part at all and can potentially lead to difficulties in understanding the explosion diagram. We therefore aim to raise the ease of notice of the exploded parts, so that they successfully stand out from their background. We will first demonstrate visualizations of single parts, before we discuss visual linking between associated parts in AR.

4.1 Part Visualization

Figure 4 shows an explosion diagram in a real world environment. The image was rendered using the techniques discussed in the previous section. It clearly shows two problems of a visualization of an explosion diagram in AR. Firstly, the exploded parts of an object are hardly distinguishable from its surroundings in its new location. Secondly, a combination of real and virtual imagery is rather confusing if presented at the same object.

4.1.1 Visual Part Discrimination

To support the visual distinction between the parts of an explosion diagram and its current video background, we highlight the bor-

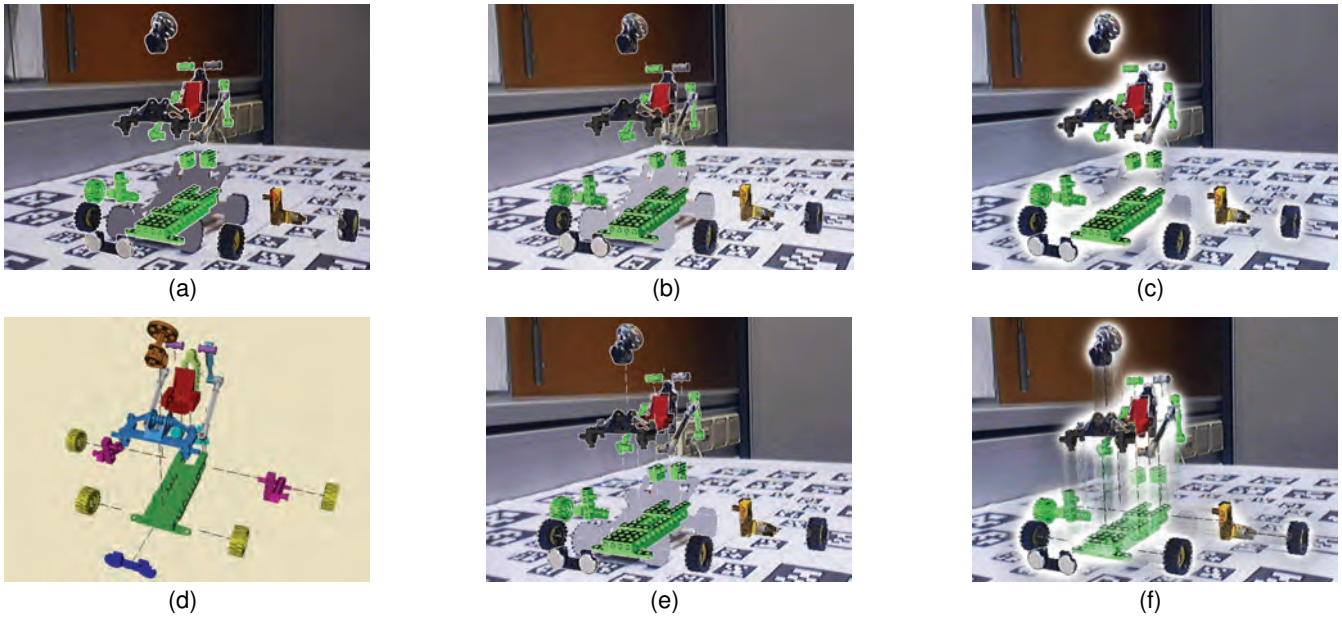


Figure 5: Visualization techniques. White object boundaries used to discriminate objects from background (a) Grey outlines are almost invisible (b) Halos and edge combination are better able to visually discriminate an object from its background. Notice, halos are only used over background information while edges emphasis the parts of an object where they overlap other parts (c) Explosion diagram using dashed lines to visually link its parts in VR (d) Explosion diagram using dashed lines in AR (e) Using motion blur in addition to visually link parts in AR (f)

der of the 2D footprint of the according part (Figure 5(a)). The border is identified in 2D by applying an edge detector on an ID buffer which is created during rendering the parts of the explosion diagram.

However, while a uniformly-colored boundary adds a very strong discriminator in situations where a high contrast between background and foreground information is given, the same techniques fails if the boundary is of similar color as the information in the background. Figure 5(a) and Figure 5(b) show the same borderline in two different level of brightness. While one is easily noticeable, the other can be hardly distinguished from the video background.

Consequently the discriminator has to be adapted to the current situation. However, while in traditional illustrations the background color is usually known, AR must deal with dynamically changing information in the background of an explosion diagram. Dynamically computing a suitable high contrast color for each object's boundary per frame creates disturbing temporal variations. It is therefore better to render the borderline using multiple shades (Figure 5(c)). We therefore compute a halo, which softly changes the brightness of an object's borderline depending on its distance to it. To not cover the explosion diagram itself with the wide spread halo, we use it only on top of the video background, while the borderline is used as visual discriminator over the parts of an explosion.

4.1.2 Visual Part Unification

The second problem of simple transformations of video texture appears when a mixture of real and virtual data appears on a single part of an explosion (see Figure 4). While an explosion diagram reveals formerly hidden parts, not enough video information is available to cover the entire visualization. Therefore the video information must be supplemented with virtual information, similar to the requirements of the restoration of hidden information.

The chosen strategy to visually unify the occurring material depends on the ratio of visible virtual to real world information. The visible pixels are counted with an occlusion query before pixel shading is applied. If the amount of available video pixel is too small (empirically set to less than 50%), we will only use the virtual color of an object (Figure 1, Figure 5). However, if enough video information is present and only some virtually shaded fragments may disturb the perception of an object, we will re-shade the virtual information to visually fit to the used real world imagery. We have implemented this re-shade operator similar to the restoration of video background, by simply computing the mean value of real world color information on the border to the virtual fragments Note the change in color on the front left wheel in Figure 4 and Figure 1.

4.2 Part Linking

To support the process of mentally reassembling an explosion diagram, traditional illustrations commonly use connection lines to visually link the corresponding parts of the object. The drawings often use rather thin and dashed (or dotted) line stylizes which must use high contrast to be perceived well (Figure 5(d)). This is usually achieved by using a homogeneously inked background and a conspicuous colorization of the connection lines, relative to the background color.

In contrast, real world environments are by no means uniform and thin lines will be easily overlooked (Figure 5(e)). Therefore, we add visual discriminators between the objects and the background imagery to emphasize the foreground objects. An example is the haloed outlines presented in the last section.

To also avoid discontinuations, we embed the virtual information within a uniformly-colored background. For this purpose we do not only use dashed connection lines, but rely on motion blur effects of the exploded parts as they form a relatively uniform visual discriminator (Figure 5(f)).

5 Layout and Animation

The previously described visualizations support the mental reconstruction of an exploded assembly by presenting visual links between its parts. To further enhance the user's ability to reassemble a model, the layout of the explosion has to be chosen appropriately to

the current task. For example, we can distinguish between layouts for exploratory and x-ray vision tasks. When exploring an assembly, the perception of its overall structure, as well as the context of the parts is of interest and therefore, the assembly can be split into single parts positioned relative to each other. Conversely, in x-ray vision a focus element is revealed in its original context. This can be described in an explosion diagram by removing entire groups of parts until the focus element is visible as single part.

Furthermore, in psychological studies on the perception of assembly plans it was found that symmetry is an important criterion for creating well structured explosion layouts [9]. Parts are symmetric, if they are of the same type and are attached to the same type of parent. For instance, the wheels of a car are symmetric and are connected to either the same type of axis or even the same axis. The layout should resemble such a symmetrical hierarchy.

Heise et. al. [9] also noticed that in a presentation of an assembly plan, symmetric parts should be presented in the same so called 'action diagram' which describes a single step out of the whole set of instructions of the assembly plan. For example, the assembly of all wheels should be presented in the same picture. We assign these findings to our animation styles which explode symmetric parts, either at the same point in time or in a row without interference of other non-symmetric parts.

To be able to fully automatically compute a layout and its animation, we identified several parameters which influence the arrangement of parts and the ordering of part removal. Firstly, an explosion sequence has to be found containing information about which parts have to be removed before other parts can safely be detached, without colliding with any of the still assembled objects. Secondly, the exploded parts have to be associated with those (parent) parts which they move relative to. Thirdly, separation directions resembling valid assembly directions as well as separation distances have to be determined. Fourthly, to be able to explode a whole group of symmetrical parts or to quickly reveal a focus element, layers (groups of partitions) are introduced.

In the implemented framework each of these parameters can be influenced independently by using a certain search strategy on the used data structure. In the following, the calculation and the influence of the parameters on resulting layout as well as on the computed animations are described.

5.1 Automatic Layout Computation

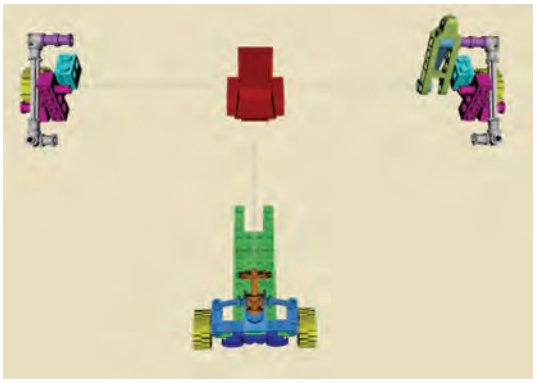
To automatically compute a layout we first have to find a valid partitioning of the object. However, to correctly disassemble an object a number of different combinations usually exist which may result in different layouts. Consequently, a strategy has to implemented to identify a single partitioning. After having identified a partitioning of an object, we have to decide which parts of it will be moved relative to others and which directions and distances are used to separate the parts.

5.1.1 Partitioning

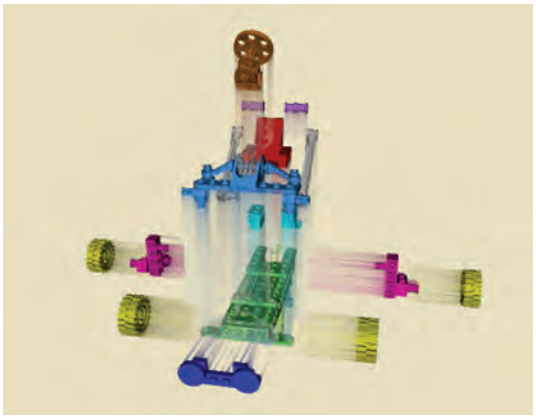
The range of all possible explosion sequences is determined by using a technique from the assembly planning domain, which follows the approach of assembly-by-disassembly. Parts or groups of parts, which are separable along their assembly direction, are identified and removed until the whole product is disassembled [10]. All potential sequences are collected in a single *AND/OR* graph which represents the assembled model in its root node and all pairs of disjoint groups of parts (further referred to as *partitions*) as children. By recursively splitting each partition into its set of pairs of valid partitions, a graph is build which contains all possible partitionings.

Following the graph from its root node to its leaves disassembles the model. However, it is very likely that an assembly can be split into several different combinations of two partitions. Therefore our system implements different strategies to choose a certain path through the AND/OR graph. For instance, a strategy used to create a symmetric layout removes the symmetric parts one after another before considering other parts. To be able to compute the layout, by using only geometrical information we identify symmetric objects by their size and their position in the AND/OR graph. Therefore, the size of the currently removed part is compared to all possibilities which can follow this part. Each subsequent part having a similar size to the current one is identified as being symmetric. The algorithm concludes the set of symmetric parts after it finds a part which is not of similar size to the current one.

A symmetric partitioning introduces a hierarchy of parts by removing similar ones from the assembly in a row. However, other search algorithms on an AND/OR graph can be found, leading to other layouts. For example, a partitioning strategy to reveal a focus object may always detect this pairs of partitions where one of them contains the most parts excluding the focus element. By recursively applying this strategy to the partition containing the focus part, a layout is achieved which reveals the focus while all other parts are being presented in a small amount of groups (Figure 6). Notice, symmetric considerations are not directly taken into account when revealing a focus part.



(a)



(b)

Figure 6: Different explosion layouts. Focused layout (a) Symmetric layout (b)

5.1.2 Part Relations

After determining the partitioning of the assembly, a relation strategy decides on which parts move relative to others by assigning parent and child relations. The decision is made on two levels. On partition-level, one partition of each partitioning is chosen to be the static parent, while the second one is the child moving relative to

its parent. The decision is further refined on part-level. Out of the set of parts being in contact between both partitions, corresponding parent and child parts are selected, thus establishing a connection between the partitions. To prevent children from being exploded relative to several different parents, a simple restriction is introduced: each partition must not contain more than one child.

This ensures that partitions already containing a child are always selected as parent partitions. Both levels can be influenced by different criteria independent from each other, thereby creating different layouts. A useful criterion can be the size of partitions and parts. For instance, good results could be achieved, when first deciding to move the partition smaller in size relative to the bigger one. Additionally, on part-level, the biggest parts were defined to be parent and child parts. In combination with the previously described symmetrical partitioning style, symmetric layouts were generated.

For focused layouts, symmetrical considerations are not of upmost importance. It is sufficient to ensure, that the focus element stays static, while the partitions are moved away from the focus element. A different approach is to move the focus element only once, if it can be separated from the rest as single part. To be able to clearly distinguish the focus from the rest, it can be exploded a greater distance than the rest.

5.1.3 Directions and Distances

To avoid display clutter by abundance of explosion directions, we restrict the overall number to only 6, which follow the object's main axis. Having computed information about groups of symmetric parts or those that focus and context relation, we can further apply this to set up distances and directions according to group memberships. For example, all objects of the same group can be set up along the same main-axis and within the same distance to their parents. Such a layout is able to visually underline the relationships among the parts.

Besides group information, the size of a part (e.g. by measuring its bounding box) may used to influence its distance of explosion, leading to a layout where smaller parts, like screws are offset a small distance from the parts they are attached to.

5.2 Computing Animation Styles

Since we are aiming for interactive presentations of explosion diagrams in AR, we are able to utilize animation styles to further enhance the comprehension of our presentations. We have implemented three different styles supporting the exploration and a focus and context visualization of an exploded object (Figure 7).

Our system allows animations which remove all parts one after another to clearly follow its assembly. To highlight symmetrical groups we furthermore implemented the functionality to show an animation of all parts of the same group at once. Nevertheless, since an explosion diagram to reveal a focus object does not necessarily require the perception of the whole assembly of the object, we have implemented a style which removes all groups at once.

6 Implementation

The system discussed in this paper runs in interactive frame rates using OpenGL and the AR framework *Studierstube* [19]. The Studierstube framework is based on the scene graph library Coin3D and it provides all the necessary components of an AR system such as tracking or video acquisition.

To be able to create a universal system, which works without manual preparation, we automatically compute the layout of an explosion by only using the geometrical information and a selection of parameters to configure the search algorithms (described in section 5). To easily create realistic animations we make also use of the physics engine ODE which is wrapped in our scene graph by using the IPSA [3] system.

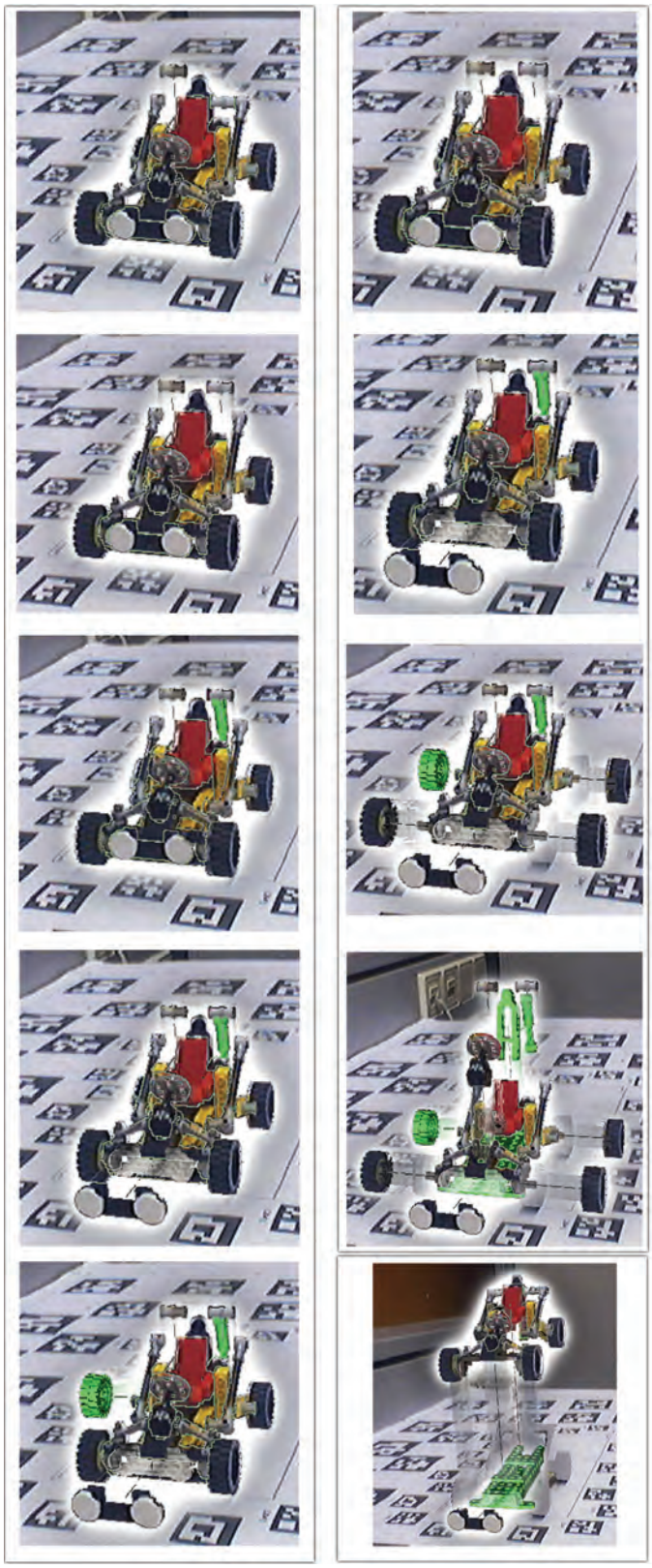


Figure 7: Different animation styles. Single part explosion (left column) Grouped explosion (right column top) All in one animation (right column bottom last image)

While the rendering of an explosion diagram in AR, including switching between different layouts or animation styles has to run in real time, the data structures are computed in a pre-processing step which last a couple of minutes depending on the number of parts of the used model. For the Lego-model shown throughout our examples it took about seven minutes and approximately 180MB of memory to compute all necessary data.

The visualizations where implemented using a deferred shading [7] environment, which consists of a set of scene graphs nodes to control which content is shaded by using which shader. Consequently, the edges are detected in image space, the applied motion blur implements the approach of Green [8] and the halo operator follows the proposed implementation of Brucker [6]. All visualization effects are implemented on the GPU using the programming language GLSL. More detailed information about the shading environment can be found in [12].

7 Conclusion

Explosion diagrams provide a powerful tool to visually communicate spatial relationships between their parts. They have been successfully applied in traditional illustrations and in scientific visualizations. We showed in this paper that they are able to be an essential visualization technique for AR applications, if care is taken in the integration.

We have shown different aspects that have to be considered when embedding an explosion diagram in a real world environment. The proposed algorithms transfer video information considering all appearing occlusions between real, relocated real and virtual objects. The algorithms can be adjusted in computational complexity depending on the needs of the AR application. To deal with deficiencies arising from relocating visual information, we have also demonstrated visualization techniques to enhance the comprehension of an AR explosion diagram. Further on, we showed a fully automatic, task dependent computation of an explosion's layout and its animation.

Our implementation is build on top of the component-oriented framework Studierstube [19]. This framework together with the fully automatic computation of layout and animations makes it possible to present AR explosion diagrams without any modifications of the underlying models being shown.

Even though this paper addresses the most important aspects for a comprehensible visualization of AR explosion diagrams, several refinements are conceivable. For example, more advanced inpainting algorithms can be used to create visually better results to compensate for lacking information after relocating real world objects. Another area of improvement are application dependent interaction techniques to modify the layout or the way its animation is presented in order to further enhance the understanding of the AR scene. Furthermore, a complete evaluation of the effectiveness of the computed layouts and their visualizations in AR is considered as future work on a series of different objects in several different real world environments. In addition, an evaluation of the influence of erroneous tracking data to the comprehension of an explosion diagram in AR has to be carried out to study the practical value of our work.

Acknowledgements

This work was sponsored in part by the Austrian Science Fund FWF under contract Y193 and the Christian Doppler Labor for Handheld Augmented Reality. We would like to thank B. Kainz for useful discussions and all reviewers for their comments and suggestions.

References

[1] M. Agrawala, D. Phan, J. Heiser, J. Haymaker, J. Klingner, P. Hanrahan, and B. Tversky. Designing effective step-by-step assembly instructions. *ACM Trans. Graph.*, 22(3):828–837, 2003.

[2] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[3] A. Bierbaum, T. Asfour, and R. Dillmann. IPSA - Inventor Physical Modelling API for Dynamics Simulation in Manipulation. In *International Conference on Intelligent Robots and Systems*, September 2008.

[4] D. E. Breen, R. T. Whitaker, E. Rose, and M. Tuceryan. Interactive occlusion and automatic object placement for augmented reality. *Computer Graphics Forum*, 15(3):11–22, 1996.

[5] S. Bruckner and M. E. Gröller. Exploded Views for Volume Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1077–1084, 9 2006.

[6] S. Bruckner and M. E. Gröller. Enhancing depth-perception with flexible volumetric halos. *IEEE Transactions on Visualization and Computer Graphics (accepted for publication)*, 13(6): to be presented at IEEE Visualization 2007.

[7] M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. In *SIGGRAPH*, pages 21–30, 1988.

[8] S. Green. Stupid OpenGL Shader Tricks. In *Game Development Conference 2003*, 2003.

[9] J. Heiser, D. Phan, M. Agrawala, B. Tversky, and P. Hanrahan. Identification and Validation of Cognitive Design Principles for Automated Generation of Assembly Instructions. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 311–319, New York, NY, USA, 2004. ACM Press.

[10] L. Homem de Mello and A. Sanderson. A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences. In *IEEE Transaction on Robotics and Automation*, volume 7, pages 228–240, April 1991.

[11] D. Kalkofen, E. Mendez, and D. Schmalstieg. Interactive focus and context visualization for augmented reality. In *Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 191–200, Nov. 2007.

[12] D. Kalkofen, E. Mendez, and D. Schmalstieg. Comprehensible visualization for augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 99(1), 5555.

[13] S. Kim, H. Hagh-Shenas, and V. Interrante. Conveying three-dimensional shape with texture. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 119–122, New York, NY, USA, 2004. ACM.

[14] W. Li, M. Agrawala, B. Curless, and D. Salesin. Automated Generation of Interactive 3D Exploded View Diagrams. In *SIGGRAPH 2008*, August 2008.

[15] W. Li, M. Agrawala, and D. Salesin. Interactive Image-Based Exploded View Diagrams. In *Graphics Interface*, pages 203–212, 2004.

[16] M. McGuffin, L. Tancau, and R. Balakrishnan. Using Deformations for Browsing Volumetric Data. In *Visualization, 2003. VIS 2003. IEEE*, pages 401–408, 19-24 Oct. 2003.

[17] A. Raab. *Techniken zur Interaktion mit und Visualisierung von geometrischen Modellen*. PhD thesis, Otto-von-Guericke Universit-Magdeburg, Germany, 1998. In German.

[18] F. Ritter, B. Preim, O. Deussen, and T. Strothotte. Using a 3D Puzzle as a Metaphor for Learning Spatial Relations. In *Graphics Interface*, pages 171–178. Morgan Kaufmann Publishers, 2000.

[19] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnação, M. Gervautz, and W. Purgathofer. The studierstube augmented reality project. *Presence: Teleoper. Virtual Environ.*, 11(1):33–54, 2002.

[20] T. Sielhorst, C. Bichleier, S. Heining, and N. Navab. Depth Perception A Major Issue in Medical AR: Evaluation Study by Twenty Surgeons. In *Medical Image Computing and Computer-Assisted Intervention*, pages 364–372, 2006.

[21] H. Sonnet, S. Carpendale, and T. Strothotte. Integrating Expanding Annotations with a 3D Explosion Probe. In *Advanced Visual Interfaces*, pages 63–70, New York, NY, USA, 2004. ACM Press.