Dynamic Compact Visualizations for Augmented Reality

Markus Tatzgern[*]
Graz University of Technology

Denis Kalkofen[†]
Graz University of Technology

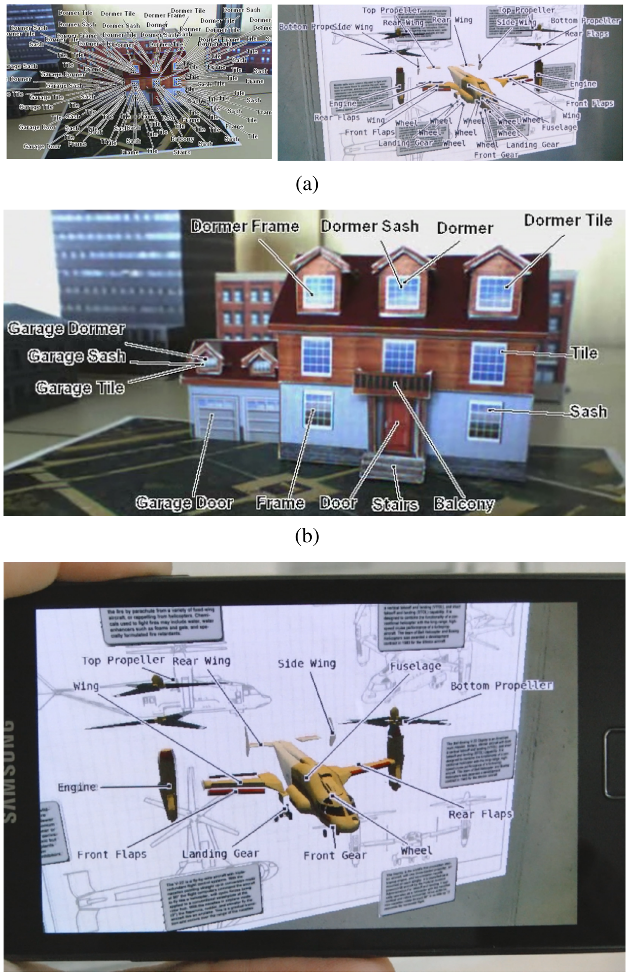
Dieter Schmalstieg[‡]
Graz University of Technology

Abstract

In Augmented Reality (AR), careless augmentations can easily lead to information overflow. Especially on small screen devices, only a limited amount of information can be displayed comprehensively. Compact visualization filters data by reducing redundancies and creating a layout of the remaining information. Previously, this approach was applied to creating static compact explosion diagrams. In this paper, we extend the approach to annotations, which are a major source of information in AR, and create compact layouts of annotations and annotated explosion diagrams. We present methods to transfer compact visualizations to dynamic AR settings and achieve interactive frame rates even on limited-resource hardware, such as mobile phones. Moreover, we create temporally coherent and scene-aware layouts.

Keywords: Augmented Reality, Visualization

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities;



(a)

(b)

(c)

Figure 1: **Compact Visualizations.** (a) Unfiltered data leads to clutter and decreases the comprehensibility of the visualization. (b) A combination of redundancy filter and layout algorithm clusters similar items and selects a single representative of each cluster. The amount of annotations is reduced, while still annotating each object class. (c) Augmentations can also interfere with elements of the real world. Exploded parts, annotations and label leader lines interfere with the text boxes of the poster. We avoid clutter and reduce interferences by filtering explosions and annotations, and by adjusting the scene layout by moving the text boxes.

1 Introduction

Augmented Reality (AR) can support users in the exploration of physical objects. A common approach in commercial AR browsers is to present information as textual or pictorial annotations to real-world objects. Such applications often rely on legacy databases or crowdsourced content, which provide a high density of data for popular subjects or locations. However, presenting such a large amount of information causes excessive screen clutter or may occlude important real-world landmarks (Fig. 1(a)).

Filter techniques reduce data to a manageable amount before a layout is created. However, filtering and layout generation are typically performed independently, which can lead to a number of problems. For instance, when using a distance based filter, the filtered annotations may cluster in only one region, thereby competing for optimal positions and degrading the overall layout. The filter output also is not stable and varies when changing viewpoint and distance. When the dataset contains redundant items, multiple objects of the same type may be annotated, and certain object classes may not be annotated at all (information loss).

We extend previous work on a combined filter and layout approach which was used to create compact explosion diagrams [11] and create compact label annotation layouts. Such compact visualizations avoid the aforementioned problems and present only a minimal number of representative items, which include all object classes without any redundancies or information loss (Fig. 1(b)).

The original approach [11] creates only static visualizations in Virtual Reality and cannot handle dynamic viewpoint changes in AR. We present a method to achieve interactive frame rates for compact explosion diagrams, compact annotations and a combination of both on typical AR viewing devices such as mobile phones. All examples are realized in AR and use real-time tracking.

Our main contribution lies in applying the idea of compact visualizaiton to dynamic AR scenes: We discuss strategies for creating temporally coherent layouts for compact explosion diagrams and compact label annotations. These strategies avoid distractions in the visualization during viewpoint changes.

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

The visualizations may also interfere with the real world scene. We present scene-aware methods to avoid these visual interferences and apply them to compact visualizations.

2 Related work

Annotation layouts for AR have been computed in a variety of ways. Coelho et al. [4] precompute layouts and place labels based on the system's tracking uncertainty either directly on the object or next to the object. Bell et al. [3] use a view management strategy [2] to place 3D labels. Azuma et al. [1] apply simulated annealing to resolve colliding elements. Rosten et al. [10] avoid covering important features in the environment by analyzing the video image.

We adopt the force-based approach of Hartmann et al. [6], because its flexibility allows us to easily define new forces influencing different layout parameters. However, compact annotations can be created by any layout technique that resolves overlaps between annotations and intersecting leader lines.

With increasing number, annotations start to compete for optimal positions and may settle in non-optimal locations. Filtering the data before layout creation allows choosing a few good locations rather than resorting to compromises. When applying a spatial distance-based filter, labels may cluster in one region, thus still competing with each other. Additionally, the filter output may vary with the viewpoint. Maass et al. [9] filter by importance derived from depth values, which leads to frequently changing elements during camera movements. Bell et al. [3] add labels based on their depth order and stop when layout constraints would be violated. Temporal coherence avoids frequent label changes. Other filters like knowledge-based filters [5] require pre-defined knowledge about the task. Hybrid filters [8] also require knowledge and may suffer from the same problems as spatial filters.

We build on a combined filter and layout approach originally developed for explosion diagrams [11]. Unlike previous work, it provides the user with a representative overview of all information without redundancies or information loss.

3 Compact Label Annotations

We apply the redundancy filter and layout combination presented in [11] to create compact annotation layouts. We annotate real world objects using their registered virtual counterparts, given as a 3D CAD model. The annotations are semantic tags, which have been assigned manually to each part of the model.

First, clusters of similar tags are determined by simple text comparison. More sophisticated methods, such as semantic networks, could also be used. The layout algorithm (in our case [6]) then positions each label of each cluster independent of the other labels to compute its interference-free optimal position. By selecting the best label of each cluster, an initial layout is created. The selection of these representative labels is based on quality criteria, i.e., the distance to the labeled part (D_P), and the size ($Size$) and visibility (Vis) of the part (see (1)). The closer a label L_i is placed to its part P_i and the higher the visibility of the part, the easier it is to understand the relation between label and part [7].

$$Q_{Label_i} = w_d \cdot D_P(L_i, P_i) + w_v \cdot Vis(P_i) + w_s \cdot Size(P_i) \qquad (1)$$

We compute visible pixels of a part by projecting it to screen space, thereby also considering occlusions by other parts. The size refers to the total number of rendered pixels, without considering occlusions. The parameters w_d, w_v and w_s control the impact of the parameters on the quality.

After creating the initial layout, the layout algorithm resolves overlapping labels and leader line intersections by changing label positions. Thus, the initially optimal positions may change. To keep the mutual influence of labels small, the initial layout is refined in an optimization step where different representatives are selected and re-evaluated. Hartmann et al. [7] suggest that labels should be placed at similar distances. Therefore, we introduce a quality criterion, which balances the distances between labels. For each pair of labels, which are direct neighbors on the bounding rectangle of the annotated model, we compute their distance $D_L(i, i+1)$. We compare it to an estimated optimal label distance D_{opt}, which depends on the projected bounding rectangle dimensions (BB_x, BB_y) and the number of labels. The quality of the entire layout is given by (2).

$$\begin{aligned} D_{opt} &= \frac{1}{n} \cdot 2 \cdot (BB_x + BB_y) \\ Q_{Dist} &= \frac{1}{n} \sum_{i=1}^{n-1} 1 - |D_L(i, i+1) - D_{opt}| \\ Q_{Layout} &= \frac{1}{n} \sum_{i=1}^{n} Q_{Label_i} + w_l \cdot Q_{Dist} \qquad (2) \end{aligned}$$

In Fig. 1(b), redundancies have been removed and only a minimal set of annotations remains that refer only to visible parts of the house. The annotations are distributed around the house. We can create combined visualizations, by sequentially executing the filter and layout approach with different data types, e.g., an annotated compact explosion diagram (Fig. 1(c)).

4 Dynamic Compact Layouts

To apply compact visualizations to AR, they must adapt to camera viewpoint changes within interactive frame rates. In addition, layouts between viewpoints must be temporally coherent, to reduce the amount of distracting changes during camera movement. Applying visualizations to AR environments can also cause visual conflicts such as mutual occlusions between the visualization and important real world structures, which must be avoided.

We present methods for creating compact visualizations which fulfill all of these requirements. We integrate some of the solutions into the combined filter and layout approach by defining additional layout optimization criteria.

4.1 Real-time visualizations using prepared layouts

The performance of the optimization for compact visualizations depends on the complexity of the input data, the layout algorithm and the quality estimation. The computation of a compact explosion diagram is computationally expensive because of the costly pixel-accurate visibility estimation and the number of optimization iterations. It takes about 30 seconds on an 2.67GHz Intel Core i7.

The performance of the optimization process can be improved by tuning the algorithms and reducing the number of iterations. However, a lower number of iterations may lead to lower quality visualizations. Additionally, when the complexity of the input data increases, the performance of the optimization decreases. Interactive frame rates for creating compact visualizations cannot always be guaranteed and even less so when considering mobile hardware with limited CPU and battery power.

To guarantee real-time performance, we precalculate compact visualizations for a discrete set of viewpoints sampled from a bounding sphere surrounding the object of interest. A virtual camera is placed at each sample point and oriented towards the center of the sphere. We adapt the distance of each camera so that the visualization always fits into the viewport as tightly as possible. At runtime, we present the computed layout which is closest to the current viewpoint of the user. To facilitate the user's perception of layout changes, the layout changes are animated over time.

4.2 Temporally coherent layouts

Strong layout changes between two viewpoints grasp the user's attention and distract from exploring the data. We reduce distractions by creating temporally coherent layouts using two approaches.

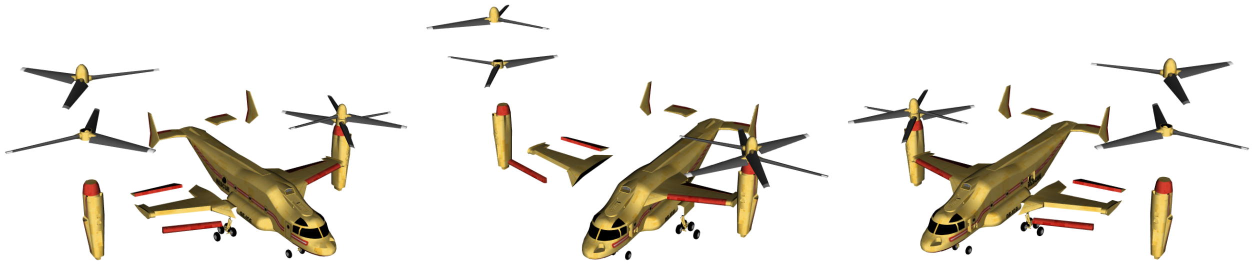
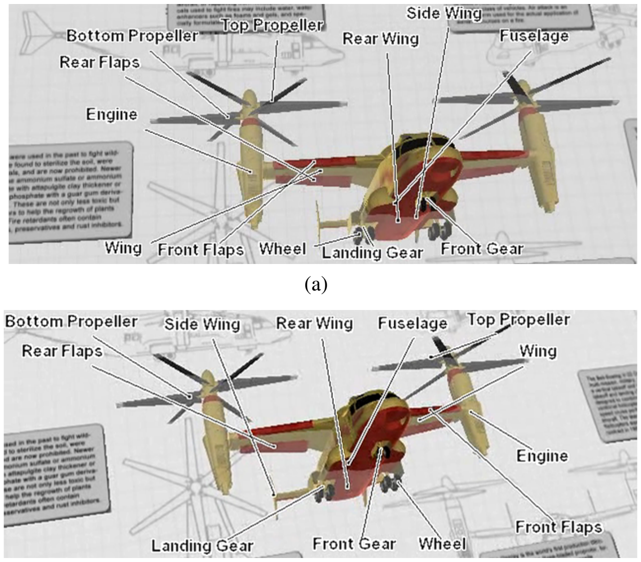

(a) (b) (c)

Figure 2: **Aligned compact explosion diagrams**. *(a)* and *(c)* show the optimal layouts for the respective viewpoints from the set of prepared optimal layouts. Note the difference between the layouts, when changing between these two viewpoints. *(b)* To reduce the amount of changes during camera motion, we switch to aligned layouts and present these during transition. Note how in the aligned layout only small changes are performed to ensure that all parts are visible. After finishing the transition, we switch to the optimal layout for the current viewpoint *(c)*.



(a)

(b)

Figure 3: **Distributing representatives**. *(a)* Fixing the label order and anchor points to the optimal compact annotation leads to large number of line intersections while the camera moves around the object. *(b)* By maximizing the distribution of annotations in the layout of the starting viewpoint, line intersections can be reduced.

Minimally different neighbors. "Aligned" layouts reduce distracting layout changes by minimizing variations between neighboring viewpoints. We define the alignment quality criterion Q_{Align}, which incorporates the difference between the current layout and the neighboring layouts. With this quality criterion, the optimization generates layouts that differ only minimally from another layout. We still consider the other quality criteria during optimization.

We achieve interactive frame rates for aligned layout transitions by preparing aligned layouts for each optimal layout, i.e., for each viewpoint. Because aligned layouts may not always represent the best layout for a viewpoint, we switch back to the optimal layout of the new viewpoint once the camera movement stops.

For compact explosions, Q_{Align} is defined in (3). The difference $Diff(L_1, L_2)$ between the exploded layouts L_1 and L_2 is given by the l_2 distance of the respective descriptors containing the positions of the parts relative to the center part of the explosion layouts. Let $P_{i,1}$ and $P_{i,2}$ be the positions of the same part in two layouts L_1 and L_2; let P_0 be the origin of the model. For each $P_{i,j}$, we calculate the vector $V_{i,j}$ from P_0 to $P_{i,j}$ and normalize it with the length of the 3D diagonal of the fully exploded model $Diag_{3D,exp}$. We calculate the difference $Diff(L_1, L_2)$ between two layouts as the l_2 distance of all distance differences between each $V_{i,1}$ and $V_{i,2}$. The more similar the layouts, the smaller the difference $Diff(L_1, L_2)$. Figure 2 demonstrates the alignment of layouts in neighboring viewpoints.

$$\begin{aligned} V_{i,j} &= \frac{(P_{i,j} - P_0)}{Diag_{3D,exp}} \\ Diff(L_1, L_2) &= \sqrt{\sum_{i=1}^{n} |V_{i,1} - V_{i,2}|^2} \\ Q_{Align} &= 1 - Diff(L_1, L_2) \end{aligned} \tag{3}$$

Minimize potential distractions. The placement of labels is constrained far less than that of explosion parts, which can only be exploded in certain directions. Even small camera motions can cause a large number of distracting changes, arising from label order changes and representative substitutions, which change the respective anchor points. It is difficult to find minimally different adjacent layouts. Therefore, we optimize the layout of the current viewpoint to minimize distractions during viewpoint changes.

Keeping the same representatives during viewpoint changes leads to changes in the label order, because leader line intersections are resolved. Keeping the label order fixed and varying the anchor points causes flickering of leader lines. Therefore, we freeze both the order of annotations and the anchor points to ensure temporal continuity. Using this approach, a large number of leader line intersections may occur after viewpoint changes (Fig. 3(a)). To minimize these distractions, we maximize the distance between labels during optimization using Q_{Dist}. We also add a quality criterion Q_{Anchor}, which implements the heuristic that anchor points lying farther apart are less prone to line intersections between the associated labels. We introduce a criterion into the optimization which maximizes the minimal distances between the projected representative anchor points A_i (4).

$$Q_{Anchor} = \frac{1}{n} \sum_{i=1}^{n} \left(\min_{j=1, j \neq i}^{n} (Dist_{Anch(i,j)}) \right) \tag{4}$$

The resulting layouts consist of elements which are less prone to leader line intersections in close viewpoints and therefore contain less distractions (Fig. 3(b)). We resolve the remaining intersections once camera movement stops by animating the transition to the optimal intersection-free layout for the current viewpoint.

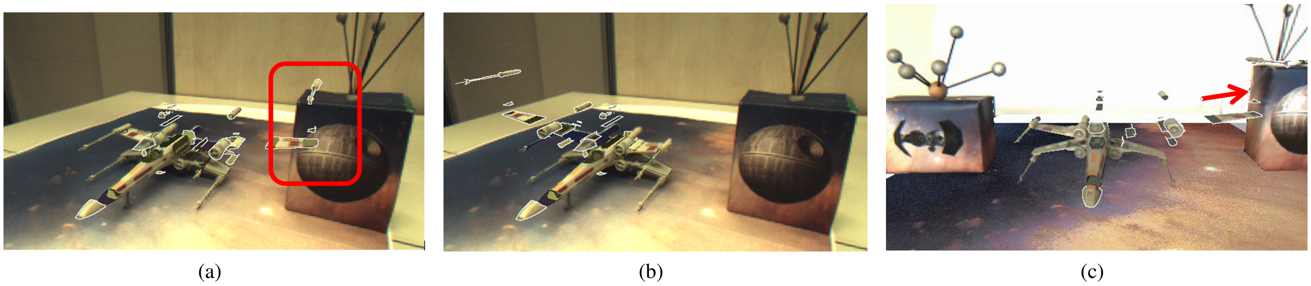


Figure 4: **Scene-aware compact explosion diagrams**. *(a)* The best layout may collide with important elements in the real world environment. By preparing a set of alternative layouts *(b)*, we are able to choose a layout which fits in the environment. *(c)* When no alternative layout fits into the scene, we displace real scene elements and can present a layout, which is a trade-off between scene modifications and layout quality.

4.3 Resolving interferences with scene elements

In visually complex environments, augmentations easily suffer from interferences with real world structures. In Fig. 4(a), the compact explosion diagram collides with the box next to the space ship, while in the combined visualization in Fig. 1(a), annotations interfere with the text boxes of the poster. We present two solutions for resolving interferences. First, we create alternative layouts to chose from at runtime. Second, if no alternative layouts is appropriate, we adapt the scene layout to accommodate the visualization. In our examples, we gather scene knowledge from registered 3D models, but it may also come from other methods such as image analysis.

Alternative Layouts. To avoid interferences with the real world, layouts must fit into the spatial constraints given by the environment. When precomputing compact layouts, we have no knowledge of real objects. Therefore, we calculate alternative layouts at each sampled viewpoint. To be able to handle a variety of scene configurations, we compute multiple layouts for a viewpoint, which vary as much as possible, using the quality criterion Q_{Alt} (5). At runtime, we select the best layout that still fits into the environment.

Starting with the best layout of a viewpoint, we iteratively optimize for n alternative layouts for this same point. A new alternative layout A_{L_n} is computed by incorporating the differences between the currently optimized layout and already computed alternative layouts of previous iterations into the optimization. This iterative optimization is achieved by sequentially performing the layout optimization n times and storing the resulting layouts as alternative layouts for the viewpoint. Fig. 4(a) shows an interfering layout, Fig. 4(b) the best alternative without interference.

$$Q_{Alt} = \frac{1}{n} \sum_{i=1}^{n-1} Diff(A_{L_i}, A_{L_n}) \tag{5}$$

Scene modification. When no alternative layout fits into the AR environment, we displace scene elements to make room for the visualization. Modifying the real world is a drastic measure and not desirable in all applications. We therefore present three different strategies to displace scene elements. First, we can use the best layout without considering the amount of modifications to the real scene, and thus may drastically change the scene. Second, we can minimize the scene modifications by choosing a less optimal layout. Third, we can find a compromise between modification and layout quality to balance both (Fig. 4(c)). Another example can be seen in Fig. 1(c), where the text boxes are moved to avoid the labels. While the first and second strategy represent the extremes of choosing either the best layout or the smallest scene modification, the third strategy covers the continuum of choices in between.

We implemented the scene modifications by extending the 2D force-based approach of Hartmann et al. [6] to 3D objects. We introduced motion constraints for real objects, to ensure scene-coherent modifications. The objects in Fig.4(c) are constrained to move on the ground plane, the text boxes in Fig.1(c) move on the wall. Constraints can also restrict how far a scene object moves.

5 Conclusion and Future Work

We applied a combined filter and layout optimization to create compact annotations, a data type frequently used in AR applications. By preparing compact visualizations from a discrete set of viewpoints, we made them feasible for AR applications running on mobile hardware with limited resources. We achieved temporally coherent and scene-aware visualizations within interactive frame rates by adding appropriate criteria to the optimization and integrating aligned and alternative layouts as well as modifying the scene layout.

A future research direction is evaluating and comparing frozen annotation layouts to the standard approach of rearranging layouts at runtime. Another avenue of research is the investigation and evaluation of scene modifications for different AR applications.

Acknowledgments

This work was sponsored by the Christian Doppler Laboratory for Handheld Augmented Reality and the Austrian Science Fund (FWF) under contract P-24021.

References

[1] R. Azuma and C. Furmanski. Evaluating label placement for augmented reality view management. ISMAR ’03, page 66, 2003.

[2] B. Bell and S. Feiner. Dynamic space management for user interfaces. UIST ’00, pages 239–248, 2000.

[3] B. Bell, S. Feiner, and T. Höllerer. View management for virtual and augmented reality. UIST ’01, pages 101–110, 2001.

[4] E. M. Coelho, B. MacIntyre, and S. J. Julier. Osgar: A scene graph with uncertain transformations. ISMAR ’04.

[5] S. Feiner, B. Macintyre, and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36:53–62, 1993.

[6] K. Hartmann, K. Ali, and T. Strothotte. Floating labels: Applying dynamic potential fields for label layout. Smart Graphics’04, pages 101–113, 2004.

[7] K. Hartmann, T. Götzelmann, K. Ali, , and T. Strothotte. Metrics for functional and aesthetic label layouts. Smart Graphics ’05.

[8] S. Julier, Y. Baillot, D. Brown, and M. Lanzagorta. Information filtering for mobile augmented reality. *IEEE Computer Graphics Applications*, 22:12–15, 2002.

[9] S. Maass and J. Döllner. Efficient view management for dynamic annotation placement in virtual landscapes. Smart Graphics ’06, pages 1–12, 2006.

[10] E. Rosten, G. Reitmayr, and T. Drummond. Real-time video annotations for augmented reality. In *Advances in Visual Computing*, pages 294–302, December 2005.

[11] M. Tatzgern, D. Kalkofen, and D. Schmalstieg. Multi-perspective compact explosion diagrams. *Computers & Graphics*, 35(1):135 – 147, 2011.