Adaptive Information Density for Augmented Reality Displays

Markus Tatzgern*
Salzburg University of Applied Sciences

Valeria Orso†
University of Padova

Denis Kalkofen‡
Graz University of Technology

Giulio Jacucci§
University of Helsinki

Luciano Gamberini¶
University of Padova

Dieter Schmalstieg‖
Graz University of Technology

(a) (b)

Figure 1: (a) A user looking for a book to read can use the glyph visualization to compare items and identify interesting books. However, finding the real world location of books is difficult because of the clutter. (b) With an adaptive information density display, the user has a better overview of relevant books.

Abstract

Augmented Reality (AR) browsers show geo-referenced data in the current view of a user. When the amount of data grows too large, the display quickly becomes cluttered. Clustering items by spatial and semantic attributes can temporarily alleviate the issue, but is not effective against an increasing amount of data. We present an adaptive information density display for AR that balances the amount of presented information against the potential clutter created by placing items on the screen. We use hierarchical clustering to create a level-of-detail structure, in which nodes closer to the root encompass groups of items, while the leaf nodes contain single items. Our method selects items and groups from different levels of this hierarchy based on user-defined preferences and on the amount of visual clutter caused by placing these items. The number of presented items is adapted during user interaction to avoid clutter. We compare our interface to a conventional AR browser interface in a qualitative user study. Users clearly preferred our interface, because it provided a better overview of the data and allowed for easier comparison. In a second study, we evaluated the effect of different degrees of clustering on search and recall tasks. Users generally made fewer errors, when using our interface for a search task, which indicates that the reduced clutter allowed them to stay focused on finding the relevant items.

*e-mail: markus.tatzgern@fh-salzburg.ac.at
†e-mail: valeria.orso@studenti.unipd.it
‡e-mail: kalkofen@icg.tugraz.at
§e-mail: giulio.jacucci@cs.helsinki.fi
¶e-mail: luciano.gamberini@unipd.it
‖e-mail: schmalstieg@icg.tugraz.at

1 Introduction

Augmented Reality (AR) browsers on mobile devices overlay geo-referenced data points directly on top of live video. A data point – or point of interest – is a three-dimensional location associated with additional information such as a textual description. Users of AR browsers not only relate items to their real world location, but can also take into account the immediate surroundings of the item. For instance, a user might identify a suitable apartement based on the provided data. However, the user notices trees in front of the apartement that block the light during the day and, therefore, removes the apartement from the list of candidates. This also underlines the benefit of AR of investigating data directly in a real world context. While a map provides a spatial reference frame, the AR view also provides ego-centric contextual information. However, when the number of data items in the field of view is large, the AR display becomes cluttered, and the user has difficulties finding relevant information (see Figure 1(a)).

Azuma et al. [1] refer to this as the problem of data density and offer two solutions. First, the amount of data can be reduced by filtering techniques. Second, view management can rearrange the data to create a more effective presentation. However, view management approaches must fail, if the amount of data grows too large. Therefore, effective filtering of the data to a sensible amount is essential for any AR browser.

A common filtering method is spatial filtering, where data points beyond a certain distance are discarded. However, this can lead to information loss, if information which is relevant to a user is removed [19]. Knowledge-based (or semantic) filtering amends this problem by selecting data based on user preferences, rather than by spatial criteria alone, and, thus, presents only relevant data [9]. However, there is no guarantee that the amount of data is sufficiently reduced to prevent clutter.

Ideally, the amount of data is reduced to an acceptable density without losing any relevant information. To this aim, we propose

(a) (b)

Figure 2: Naive spatial clustering helps only temporarily. (a) The user's display has an acceptable amount of clutter, but the user unfolds an item. (b) The display becomes overloaded.

to aggregate related data points by clustering, rather than removing data with a filter (see Figure 1(b)). Clustering has the advantage over filtering that the complete information space is preserved. State-of-the-art AR browsers[1] cluster data points by spatial proximity in 2D image space to reduce visual clutter by showing only representatives of the grouped items. To avoid visual clutter for a large data set, the clusters must contain a large number of items. Unfolding such clusters during interaction with the data re-introduces clutter (see Figure 2).

We address this issue by creating an information hierarchy, which is conceptually similar to semantic level of detail [8]. By recursively applying clustering, an information hierarchy is built. Our clustering approach not only considers user-controlled spatial attributes (e.g., distance), but also non-spatial attributes (e.g., semantic tags). The sum of these user-weighted attributes provides a ranking of the data, which expresses its relevance to the user. To avoid visual clutter, a display algorithm shows data which is relevant for the user in more detail, while it always adapts the overall amount of information to the available image space. It does so by solving an incremental optimization problem, deciding which nodes in the hierarchy are selected for display. Users can dynamically adjust priorities to interactively drill down on data deemed relevant, and reveal all available details on demand.

Generally, application cases for the presented method include exploratory search tasks, in which a user wants to get an overview of the available data, before making an informed decision. For instance, various shopping scenarios involve this kind of task. A customer in a video or music store would like to browse the available selection based on preferences, without removing any items beforehand. A parent may want to look for toys in a shop based on the preferences of the child. In a food store, a customer, who is unsure about the upcoming dinner, states food preferences and chooses accordingly. A researcher in a university library looks for related work.

To the best of our knowledge, this paper presents the first view management system which uses adaptive clustering rather than filtering, yielding the following improvements over previous work:

- Our approach condenses information rather than discarding it; all data remains directly accessible.
- Clutter is reliably avoided, since per-frame optimization suppresses excessive data density.
- User-controlled combination of both spatial and semantic criteria for clustering with priorities yields a *ranked* selection of data points.
- Adaptive selection of the data which is most relevant for the user is possible with both implicit interaction (free viewpoint exploration) and explicit interaction (data point selection for drill-down).
- Our technique can work with both view-dependent and view-independent attributes, and ensures suitable temporal coherence in both cases.

Thus our system is the first first-person AR system that combines the advantages of ranked search and free viewpoint exploration.

[1] https://help.here.com/de/wp8/citylens

2 Related Work

The two most common techniques that are used to reduce visual clutter are filtering and clustering. While filters completely remove items from the presentation, clustering algorithms group them based on a given distance function that expresses the similarity of items.

2.1 Filtering

In AR, augmentations can be filtered by spatial or semantic criteria. A spatial filter is often implemented as a magic lens [4, 15], which filters information in screen or object space. Spatial filters typically require an undesirable amount of user interaction to investigate the data in its entirety. In addition, such spatial filters only work locally in a small region and, therefore, do no create overviews of the input data, which provide a general picture of the overall available data. In contrast, the approach presented in this paper permits the user to explore details and, simultaneously, provides a general overview.

Feiner et al. [9] present a knowledge-based filter, which filters information based on knowledge about the user's task. For instance, such a filter can show the current step in a sequence of assembly operations based on the status of the overall assembly. Knowledge-based filters are the logical choice for showing sequences, but can also be used to visualize larger data sets. For this purpose, a degree-of-interest function can be defined that filters data based on priorities set by the user [10]. We take a similar approach by classifying data based on an interest function. However, int our approach, data that is not relevant to the user is not lost completely, but aggregated into an abstract representation.

Location-based services often combine spatial and knowledge-based filters [21]. Julier et al. [12] present such a combined filter for AR. Each data item has a spatial location and a surrounding region of interest, which is scaled based on these priorities. Only when the user enters a region of interest, the data is presented. More interesting items have a larger region of interest and, therefore, appear earlier in the AR view. This approach can still lead to visual clutter, when the user enters the region of interest of a large number of items. Our approach always controls the amount of clutter.

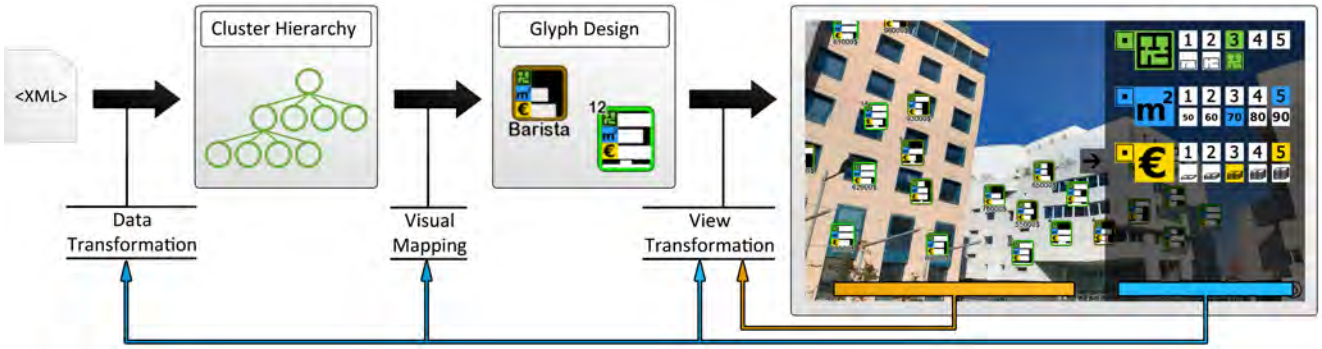

Figure 3: System overview. Our information density display follows the information visualization pipeline. In the data transformation step, the input data is clustered to create a hierarchical representation of the data. The data items of the hierarchy are encoded as glyphs. The user can influence each step explicitly via the user interface (blue line) or implicitly by changing the viewpoint and manipulating the glyphs directly (orange line).

2.2 View management

View management algorithms automatically arrange augmentations on the screen, so that they avoid overlaps between each other and the world. With an increasing number of augmentations, it becomes harder for view management algorithms to arrange the items on the screen. Maass et al. [13] and Bell et al. [3] handle overload in view management by omitting excessive annotations. Maass et al. [13] derive importance from depth values and can choose to omit labels beyond a certain distance. However, this approach is not temporally stable during camera movements. Bell et al. [3] filter annotations based on the visibility of the referred elements and a user-defined importance value. Their algorithm arranges annotations with the highest priority first and stops adding annotations, when layout constraints would be violated. This approach results in more stable layouts, but does not scale to large data sets. In contrast, our approach makes sure that all relevant information is preserved.

2.3 Clustering

In information visualization, hierarchical clustering can be used to reduce the visual complexity, e.g., of graph visualizations [2, 11]. Such aggregations require visual abstractions that present the data at an appropriate semantic level of detail.

Tatzgern et al. [18, 19, 20] present a combined filter and layout approach that filters data based on redundancies to reduce visual clutter. First, data is clustered by similarity. A selection algorithm selects only representative items from redundant clusters. This approach can only avoid clutter, when there are redundancies to exploit, and is not sensitive to the amount of used screen space. Our work uses clustering from semantic attributes and respects the desired screen space density.

Our approach is inspired by constant density information displays for map visualization. Such displays fill the available screen space with a constant amount of information, even during zooming. Woodruff et al. [24] use a regular grid in screen space as a measure of local clutter, but do not address temporal coherence. Dix and Ellis [6] use random sampling to select data items, such that their locations are also balanced over the available screen-space. However, this approach selects different items in every run. Both approaches filter by removing data from the set. Our approach fills the screen with items representing a cut through a cluster hierarchy. We dynamically fill the screen with items, while avoiding clutter locally around an item. Because we avoid a grid or similar quantization, we naturally achieve temporal coherence of the visualization.

3 Overview

Our AR browser lets users query geo-referenced data about their surroundings, such as restaurant information or real estate offers, from online databases such as Google Places[2]. The data points are visualized as annotations in an AR view. A typical use situation will involve several hundreds or thousands of data points, more than can be presented in full on the screen, making view management necessary.

Visualization for AR browsers has similar requirements as in classic information visualization, which involves the three main stages of the information visualization pipeline [5]: data transformation, visual mapping and view transformation (see Figure 3).

In the **data transformation stage**, a cluster is created from the data points. The clustering is based on similarity among the data points' attributes. Each attribute has a user-defined weight.

In the **visual mapping stage**, every cluster point is transformed into a glyph showing a summary of the most important attributes.

In the **view transformation stage**, a 2D display is generated from a selection of cluster points. The selection is based on the user's current 3D viewpoint and the user's interactive selections. The user can change the selection interactively by setting desired values for each attribute or by explicitly unfolding parts of the cluster hierarchy (with a tapping on the items on the screen). The selection is dynamically computed to best match the user's expressed interest, while respecting a maximum point density on the screen.

In the following sections, our approach is discussed in more detail.

4 Hierarchical clustering

In the data transformation stage, a hierarchy of clusters is computed based on similarity of data points. A flat partitioning would require knowing the number of clusters in advance and cannot reflect the structure of the data well [14, p.377]. Instead, by using a cluster hierarchy, the view management can later decide for every frame at which level the hierarchy should be cut and, therefore, how many clusters should be presented.

We consider a set of data points $\mathbf{D} = \{D_i\}$, each with a set of attributes $\mathbf{A}$. We denote the attribute set of D_i as $\mathbf{A}_i = \{A_{i,j}\}$. Each attribute can have an arbitrary data type, describing aspects such as user satisfaction rating, social tags or pricing. Two attribute values $A_{i,j}$ and $A_{i',j}$ of the same type $\mathbf{T}$ can be compared with with a comparison function $cf_j : \mathbf{T} \times \mathbf{T} \rightarrow [0,1]$, which yields 1 if two values are identical, and 0 if two values are most dissimilar.

[2]https://developers.google.com/places/

The user expresses the desired information by setting desired values $\mathbf{U} = \{U_j\}$ and weights w_j for each attribute ($\sum w_j = 1$). A weight of 0 indicates that the user does not care about a particular attribute; in this case, the desired value for this attribute can be arbitrary. With the comparison functions, we can compute the similarity $S : \mathbf{A} \times \mathbf{A} \rightarrow [0,1]$ of two attribute sets as the weighted per-attribute difference:

$$S(\mathbf{A}_i, \mathbf{A}_{i'}) = \sum_j w_j \cdot cf_j(A_{i,j}, A_{i',j}) \quad (1)$$

Once the user has set weights and desired values, the clustering algorithm can be started. We use top-down divisive k-means clustering to create our hierarchical cluster tree composed of nodes $\mathbf{N} = \{N_i\}$. All data points initially form a cluster that corresponds to the root of the tree. We recursively perform k-means on the root, until the leaves of the tree correspond to the smallest possible cluster containing only one item, i. e., $\mathbf{D} \subseteq \mathbf{N}$.
K-means creates clusters based on the similarity function S. We define the branching factor of the tree by choosing a number of clusters for each iteration of k-means. For our experiments, we set this factor to four.

Both position P_i in world space and position p_i, the projection of P_i to screen space using the current camera position, may, but need not be used as attributes. Using only P_i as an attribute and the Euclidean distance (either in 3D or just geographic longitude/latitude) as the comparison function will result in a conventional POI clustering based on geographic proximity. Such a clustering is *view-independent*, i. e., computed in world space and not in screen space. This has the advantage of a temporally coherent view management.

World-space position P_i can easily be combined with other attributes using appropriate weights. Consider a user searching for restaurants. The type of cuisine (Italian, French, Chinese) may be given a high weight. In this case, restaurants with the same cuisine may be clustered first, even if they are not lying closely together. However, restaurants that are very far apart will not be clustered, even though they offer the same cuisine. Setting the weights appropriately allows the user to balance her needs or preferences.

The world-space position of an intermediate node is computed as the centroid of the children's positions. When computing intermediate notes, we found it useful to decrease the weight of P_i proportional to the graph distance of a given N_i from its leaves. In this way, position has a stronger influence on clustering in the lower, more "concrete" layers of the tree, but similarity of semantic attributes has a stronger influence in the upper, more "abstract" layers of the tree. This means that objects from the lower layers will be projected onto close-by point in screen space. Replacing these points with their centroid only minimally changes their placement on the screen and, thus, retains good spatial accuracy. Intermediate nodes from high layers may be placed further from the natural screen-space position of the represented leaves. However, node from high layers represent more abstract information with a general indication of direction, such as "all Italian restaurants to the west".

The comparison function may involve operations that depend on a user's situation and are, thus, subject to change as time and space change. For example, we can use a mean or maximum value for normalizing a certain attribute, making the comparison function dependent on the current database population. We may also consider a routing algorithm that determines the time to walk or drive to each destination from the user's current position. Such estimates may become increasingly wrong over time. For such time-varying attributes, we trigger re-computation at certain intervals, for example, every few seconds, or if the user has moved a certain distance.

5 Optimal Label Selection

In the following, we describe the view transformation stage of our approach to balance the available screen-space against visualizing relevant data to the user. The system must also handle viewpoint changes in a temporally coherent way. A view management algorithm makes sure that any interfering data items are rearranged.

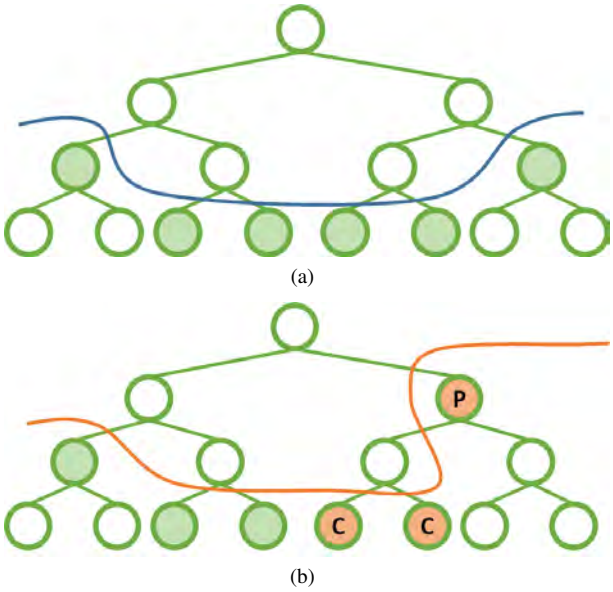

Figure 4: Selection from the cluster hierarchy. (a) A greedy best-first search creates a a cut through the tree (blue line). Nodes below the cut, shaded in green, are placed on the screen. (b) The cut must either place all children or no child on the screen. The orange cut is invalid, because the predecessor (P) of two nodes (C) is included.

5.1 Initial label selection

We want to select a set of labels $\mathbf{L} = \{L_k\}$ representing a cut through the cluster tree, so that all data points have some representative. We write $children(i)$ for the set of all direct children of N_i, $children^*(i)$ for the set of all direct and indirect children of N_i (including N_i itself), and $leaves(i) = \{x | x \in children^*(i) \wedge x \in \mathbf{D}\}$. Using these definitions, we can describe the set of all possible cuts as follows (see Figure 4):

$$cut(\mathbf{N}) = \{\{\mathbf{L}\} \mid \mathbf{L} \subseteq \mathbf{N} \wedge L_k \not\subseteq children(L'_k) \; \forall \; (L_k, L'_k) \in \mathbf{L} \wedge (\exists \; L_k \in \mathbf{L} \mid D_i \in children^*(L_k) \; \forall \; D_i \in \mathbf{D})\} \quad (2)$$

With these considerations, we can select a suitable $\mathbf{L} \in cut(\mathbf{N})$ for a given user position. By introducing a cost and benefit metric, we can interpret the label selection problem as a constrained optimization problem. It tries to fill the screen with the most beneficial labels, by optimizing a benefit function $B(k)$

$$\max_{\mathbf{L} \in cut(\mathbf{N})} \sum_{L_k \in \mathbf{L}} B(k) \quad (3)$$

while avoiding excessive clutter by respecting a maximum cost $C(k)$

$$C(k) \leq C_{max} \; \forall \; L_k \quad (4)$$

The benefit of a leaf D_i is given by its similarity to $\mathbf{U}$, i. e., $S(\mathbf{A}_i, \mathbf{U})$. The benefit of an intermediate node depends how well it can represent its leaves. We account for this fact by weighting the benefit with the label's spatial displacement in screen space, w_P, and the semantic similarity of the data points represented by the label,

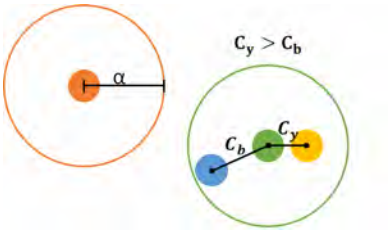

Figure 5: To measure the visual clutter of a data item, we consider a region with radius α around the item (red circle). The blue and the yellow node fall within the radius of the green node. The cost for placing blue is smaller, because it is farther away from the green node and produces less clutter.

w_S. The spatial displacement w_P gives more benefit to intermediate nodes, which are close to their data points, expressed as relative inverse distance:

$$w_P(k,k') = \frac{1}{1+||p_k - p_{k'}||} \tag{5}$$

The semantic similarity w_S gives higher benefit to an intermediate node representing homogeneous data points, which have a high average similarity S:

$$w_S(k) = \frac{2 \cdot \sum_{(L_i,L_{i'}) \in leaves(k), i \neq i'} S(\mathbf{A}_i, \mathbf{A}_{i'})}{|leaves(k)| \cdot (|leaves(k)|+1)} \tag{6}$$

We combine these terms in a recursive definition of a benefit metric $B(k)$:

$$B(k) = \begin{cases} S(\mathbf{A}_k, \mathbf{U}), & \forall D_k \in \mathbf{D} \\ w_S(k) \cdot \sum_{L_{k'} \in leaves(k)} (B(k') \cdot w_P(k,k')), & \text{otherwise} \end{cases} \tag{7}$$

The cost of including a node N_k in $\mathbf{L}$ is related to the clutter it produces. Using w_P, we can express the clutter as local density of other labels in a neighborhood of radius α around a node (see Figure 5):

$$C(k) = \sum_{N_{k'} \in \mathbf{L},\ |p_k - p_{k'}| < \alpha} w_P(k,k') \tag{8}$$

This optimization problem can be approximated with a greedy best-first search (BFS) [16]. It starts with the root of the cluster tree and keeps propagating the cut through the tree towards the leaves, by unfolding an intermediate node $L_k \notin \mathbf{D}$, i. e., replacing L_k with its children. The unfolding operation changes the relative benefit $R_u(k)$:

$$R_u(k) = \left(\sum_{L_{k'} \in children(L_k)} \frac{B(k')}{C(k')} \right) - \frac{B(k)}{C(k)} \tag{9}$$

The $R_u(k)$ are kept sorted in a joint queue with decreasing order. In every step, the node with the highest relative benefit is chosen, provided it is eligible, i. e., $C(k') \leq C_{max}\ \forall\ L_{k'} \in children(L_k)$ for unfolding. This process terminates, if no more improvements can be found.

Greedy BFS quickly converges towards a useful result, but can get stuck in a local minimum. We therefore refine the BFS result with a random search approach based on threshold accepting [7]. Threshold accepting applies small random changes to the solution and temporarily accepts solutions that are worse than the current best

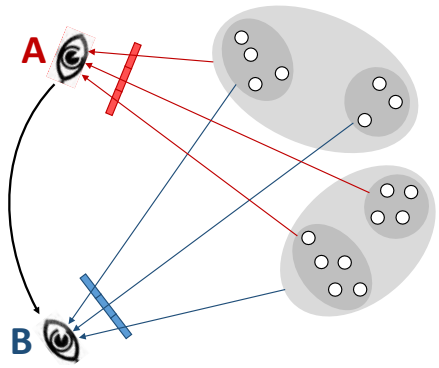

Figure 6: Data points (white discs) are clustered hierarchically (grey bubbles). For every viewpoint, an optim al selection of cluster nodes is produced, which fills the available space on the screen. When the user moves from A to B, the selection of clusters changes automatically.

Figure 7: Since the selection from the cluster is computed in panoramic coordinates, a head rotation ($A \rightarrow B$) does not require a re-clustering; only a translation ($B \rightarrow C$) does.

configuration. We randomly select an unfolding and a folding operation (replacing a group of nodes by their common parent). The quality of the resulting configuration is determined as usual, via the cumulative benefits. Note that during this optimization step, the maximum cost given by equation 4 is not exceeded. The optimization terminates after a defined number of iterations, which makes performance very predictable.

5.2 Temporal coherence

After the initial selection, the labels are presented to the user. For interactive use, it is important to ensure temporal coherence and suppress jumping motion of labels. Therefore, labels are adjusted incrementally in every frame. After a change of viewpoint, the p_k are recomputed, and the queue containing the R_u is re-sorted accordingly. BFS is restarted on the re-sorted queue. Usually changes are small and continuous, so the optimization converges quickly after only a few operations.

Note that in the most common cases, the view transformation does not require a re-computation of the k-means clustering and can run with camera frame rate (30Hz) on mobile devices. Re-clustering is only necessary if user location dependent attributes change significantly, or if the user changes the attribute weighting.

Many attributes considered in this process are not dependent on the user's current location, for example, the Euclidean distance of two locations or the difference in real estate prices between two locations. In these cases, any viewpoint change simply requires adjusting the selection (Figure 6).

Re-clustering may not be necessary, even if attributes are used which depend on the user's current location, such as the walking distance to a location or the visibility of an item from the user's current viewpoint or not. Since users will often stand and look around, we compute the view transformation in panoramic coordinates centered at the user's current location. Updating the selection for a user who just turns the head and does not (or only minimally) move (Figure 7, A to B), does not require any re-clustering. Likewise, the

user may dynamically adjust the desired attribute values without triggering a re-clustering.

Only if the user walks by more than a certain distance (Figure 7, B to C) (usually a rare event), a re-clustering is necessary. The computation takes below one second in our experiments (Table 1), which is usually not disturbing for the user.

5.3 Local view management

Assigning a fixed position to labels turns label placement into a discrete label selection problem: We only need to determine the set of labels chosen for display. However, this discretization can lead to poor results, if many important labels occupy the same region.

We increase the quality of view management after label selection by subjecting them to another optimization that purely considers spatial placement. This placement employs the "hedgehog labeling" technique [17], which places annotations in world space to achieve stable layouts. The movement of a label is constrained to a plane parallel to the image plane. The scale is set up so that annotations have the same size, independent of their distance. This approach gives enough flexibility to compensate for poor initial placement, while ensuring temporal coherence of label adjustments. We use hedgehog labeling both when a node is first displayed and to compensate for local clutter after a viewpoint translation. However, we must handle the case where labels move off the screen. A leaf node will simply be omitted, but an intermediate node representing at least one data point on the screen must always be displayed. This problem can be handled by the hedgehog labeling by adding a constraint enforcing that only on-screen coordinates are eligible.

6 Glyph Design

The visual mapping stage transforms data points into glyphs encoding the relevant attributes. Glyphs are a common way to visualize multi-dimensional data in a meaningful way [23]. In our case, glyphs inform the user about the represented data points and the relation of the different categories to the current user preferences.

It also should have a compact visual footprint, so that it does not cover too much screen real estate. We use two variants of the glyph, one for leaves (individual data points) and one for intermediate nodes. The glyph for a leaf L_k (see Figure 8(a)) should convey the relevance of the represented data point to the user directly. It consists of a square icon with a footer text describing the data point (e. g., stating a business name). The icon has a thick frame, which is color-coded according to the leaf's benefit $B(k)$, where 1 is green and 0 is red. Inside the square, there is room for up to three attributes selected by the user, arranged as a horizontal mini-barchart. Next to an icon identifying the attribute, the agreement of the selected attribute with index j to the user's preference is shown, i. e., $cf_j(A_{k,j}, U_j)$.

The glyph for an intermediate node (see Figure 8(b)) summarizes the relevance of the leaves it represents. It is also a square icon. The number of leaves represented by the glyph is shown in the top left corner, similar to icons of popular mobile user interfaces. Like in the leaf glyph design, the frame of the glyph is color coded to show the average benefit of the contained nodes. A box extending on a line at the bottom of the glyph shows the spatial extent w_P of the cluster relative to the screen width, depicted as a fraction of the glyph width. Similar to the leaf glyph, a mini-barchart inside the square displays up to three selected attributes. However, the bar size for each attribute is proportional to the average agreement $avg(k, j)$ over all leaves with the user's preference:

$$avg(k,j) = \frac{\sum_{L_i \in leaves(k)} cf_j(A_{i,j}, U_j)}{|leaves(k)|} \tag{10}$$

Averaging the content of the node provides a good general overview of the contained data values. However, when users look for the best matches to certain criteria, the averaging operation hides potentially

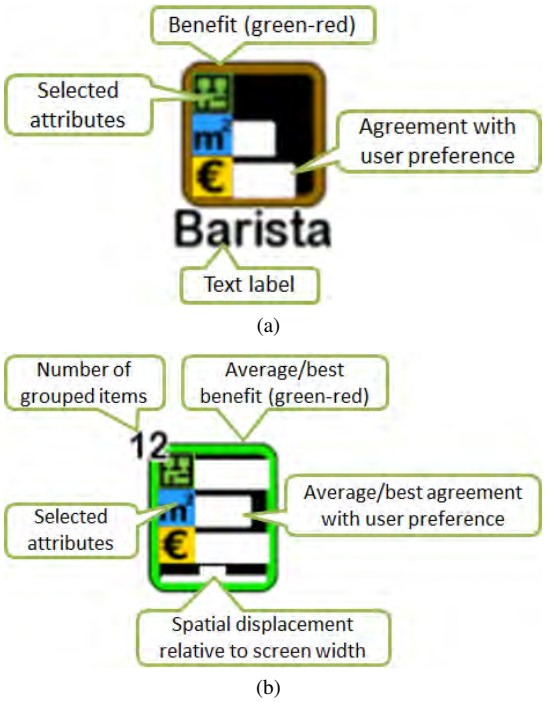

Figure 8: Glyph design. (a) The glyph for a single data point compares the data to a user defined reference value. The border color indicates the average matching quality of the selected attributes to the reference attributes (the greener the better). (b) The glyph for grouped items shows the information averaged over all items.

good matches in the intermediate node visualization. Alternatively, an intermediate node can represent the leaf node with the best benefit and adapt its appearance accordingly.

7 Interacting with Clusters

We allow user interaction in every step of the pipeline of Figure 3. To be able to change the structure of clustering during the data transformation, we provide the users with an interface for adapting the weights of the attributes and, thus, changing their preferences. To facilitate the interaction, the interface allows users to specify the weights not as absolute values, but relatively to each other. Internally, the relative settings are mapped to weights that sum up to one, as required by the algorithm. Changing the weights triggers recalculation of the hierarchy.

The user can change the weights of the algorithm by using the numbered buttons of the user interface as shown in Figure 3. The user interface shows three attributes. The weight w_j for each attribute is calculated based on the integer values V of the buttons. Each weight w is calculated according to the following equation, which makes sure that the sum of the calculated weights is one.

$$w_j = \frac{V_j}{\sum_{i=1}^{n} V_i}$$

The user interface also allows users to specify the user preference values U that are used to calculate the benefit of the nodes. Consequently, this changes the selection of nodes from the hierarchy, but not the hierarchy itself. The visual mapping of the glyphs is also updated accordingly.

Users set the preference values U using the row of symbols beneath the numbered weight buttons. We chose this interface to simplify the selection of preference values. Instead of using a slider or entering values by hand, the user simply taps a symbol. Each

(a) (b)

Figure 9: Interacting with groups. A user can unfold grouped items by tapping their glyph. (a) The user taps the glyph indicated by the red arrow. (b) The unfolded new elements are highlighted using a blue outline. To make room, other items are replaced by the group they belong to.

#POI	Clustering	Selection	Sum
100	63ms	16.5ms	79.5ms
500	258ms	59.9ms	317.9ms
1000	744.4ms	111.1ms	855.5ms

Table 1: Performance data. The table shows performance measurements (in milliseconds) of the clustering and the selection steps. The application code was not optimized and was executed in a single thread on an Intel i7 with 2.7GHz.

symbol represents a distinct value that has been set up beforehand, thereby building an ordinal scale of values. A user can select one of these values as preferred value for this attribute. Internally, the system normalizes the range of values and calculates their similarity using the Euclidean distance.

In the view transformation step, the user must be able to manually unfold clustered representations. We allow the user to drill down by unfolding the subsequent levels of the hierarchy step-by-step. The user simply clicks on a glyph representing an intermediate nodes to unfold the next level of the hierarchy. If the user's unfolding leads to a violation of the clutter metric, the system will first try to relax the situation locally via hedgehog label adjustments. However, it may occasionally be necessary to invoke fold operations on other labels to make room for the user's unfolding (see Figure 9). This problem can be handled implicitly in the label selection optimization by assigning a higher weight to the benefit of the label unfolded by the user.

By default, this user-driven benefit will slowly wear off with an exponential decay. This allows the user to explore different areas on the screen or branches of the cluster hierarchy incrementally. Older user interactions will become less relevant over time and eventually make room for newer interactions. However, the user may instruct the system to remember choices indefinitely.

We measured the scalability of the system and, thus, of the interaction on an Intel i7 with 2.7GHz. Note that the application used only a single thread and the code was not optimized. The results are shown in Table 1. Even when working with 1000 data points, our system could perform the clustering and selection steps in less than one second. Given that AR browsers usually work with remote data, these steps could also be performed on dedicated servers.

8 User Evaluation

We conducted two user evaluations: The first one was a pilot study in which we compared our interface against a conventional filtering interface. In the second one, we explored the effect of different degrees of clustering.

8.1 Comparing to Filter Interface

In this pilot study, we performed a qualitative evaluation to compare our hierarchical clustering interface against a conventional filtering interface. We were interested in the user's qualitative feedback comparing the two interfaces: our interface versus one resembling commercial AR browsers. We concentrated on the magnitude of the potential improvement of the user experience and not on quantitative aspects such as task performance. Instructing users to "work as fast as possible" may, in fact, influence the participants to adopt a behavior that does not resemble a typical use case anymore. We, therefore, instructed users to work at their own pace and favor insight over speed. Note that the second study, explained in the next section, takes a closer look at the mechanics of our user interface.

We investigated two interface conditions: our hierarchical clustering user interface (**HUI**) and a filter user interface (**FUI**). The folding and unfolding of HUI behaved as described before. We also allowed participants to switch the presentation of intermediate nodes between showing the average of all contained items and showing only the best contained item (see Section 6). FUI behaved like a state-of-the-art AR browser filter interface[3]. The data items were filtered according to parameters set up by the user. As common in AR browsers, the data items were presented using a simple glyph representation, consisting of a circle and additional textual information, with full details revealed on demand. Setting filter parameters removed information from the AR view that did not correspond to the filter parameters. We integrated the same view management system into FUI as we use in HUI, so that occlusions between glyphs could be resolved.

The interface condition was counterbalanced among the participants. We used the same amount of data in both interfaces, but changed the attributes of the data between conditions.

Hypothesis. FUI removes data that does not correspond to the user preferences. However, the screen will be cluttered, if too many data items are preserved. In addition, setting the filter parameters in FUI to find relevant data points might be challenging. HUI aggregates items and reduces clutter. HUI allows users to set preferences to which items are compared to, which makes it easier to identify relevant data items. Therefore, we expected HUI to be preferred.

Scenario and Setup. We used an accommodation search scenario in the evaluation. We gave the participants the task to find rental apartments that fulfill certain requirements. For this purpose,

[3] www.layar.com, www.wikitude.com

	Mode		Mean (SD)	
Question	FI	HI	FI	HI
I liked the visualization of data items in the interface.	3	1	3.375 (0.92)	1.375 (0.74)
I found the visualization of data items helpful.	3	1	3 (0.53)	1.25 (0.46)
The interface was convenient for finding apartments.	2	1	2.625 (0.74)	1.5 (0.76)
The interface was convenient for comparing apartments.	3	2	3.5 (1.07)	1.75 (0.71)
The interface provided a good overview of the data.	3	1	3.125 (0.99)	1.75 (1.04)

Table 2: Questionnaire results for first study. The mode and mean (with standard deviation) of the questionnaire results of a five-point Likert Scale (1 .. strongly agree).

we performed the study outdoors in an area with high-rise apartment buildings. The attributes of the data were created randomly and registered to the locations of the apartments. Their locations were not occluded and, therefore, clearly visible from the participants' current position and registered to real world objects. We did not use hidden objects, since the distinction between visible and hidden would complicate the visualization and was not part of our research question. The study followed a within-subject design.

We used the following apartment attributes: number of rooms (scale with three entries), square meters (scale with five entries), price range (scale with five entries). Participants could set one or more of the categories of each attribute. The text of the leaf node corresponded to the final price of the apartment.

We deployed the interfaces on a tablet computer (Microsoft Surface Pro 2, Windows 8.1) and used the front-facing camera for capturing the surroundings. We used a panorama tracker [22] and device sensors (IMU) to determine the orientation of the device and align the data with the real world. The resolution of the application was set to 1280x720 and corresponded to the camera resolution.

Task. In order to see how participants would use the interfaces, we gave them an open task for apartment search. We asked them to look for apartments that suited their criteria. The underlying motivation is that users identify interesting apartments and collect them into a list to revisit them later to allow better decision making.

Procedure. We met the participants at a meeting place, where they filled out the consent form and a demographic questionnaire. Then we moved to the apartment site, where the study was performed. At the site, we explained the first interface.

After they were confident with using the interface, we asked them to solve the given task. We also asked the participants to speak aloud during the task. After finishing the task with both interfaces, they filled out a questionnaire asking for feedback and rating the interfaces. We concluded the session by asking open questions regarding their experience.

Results and Discussion. A total of 8 people (3 female) aged 26–34 ($mean$=31.5, sd=3.17) took part in the study. In the questionnaires, we forced participants to decide for either FUI or HUI as the preferred interface. Seven of eight participants preferred HUI. An exact binomial test found a significant difference ($\alpha = 0.05$) that HUI is the preferred choice ($p < 0.05$). In addition, we asked participants, if the intermediate node in HUI should show the average of all or the best item. All participants preferred the best item, because finding the best item is most relevant for a search task. The average would hide this information. In general, the questionnaire revealed that participants were in favor of our interface (see Table 2).

The one participant who did not prefer HUI argued that while the clustering reduces the amount of clutter, the registration of the items summarized by the cluster is lost. In FUI, the location of an item was clearly visible, if the amount of clutter was not too high. Therefore, the participant suggested grouping items by stronger location-based criteria, such as the floor number, and also adding more options to the user interface for targeting items based on their spatial location to the user interface. Note that while we did not include stricter location-based groupings in our study, our system can easily support this by adding the respective attribute to the data.

In general, 62.5% of the participants made use of the real world registration of apartments during their search for an apartment. For instance, if several apartments had attributes of similar quality, the one that was on a higher floor was preferred. This underlines the usefulness of the spatial registration of items for this task.

Participants preferred HUI, because it provided a better overview of the data. In HUI, 75% of the participant not only considered the best matches, but also checked for other apartments that were close to the set criteria. FUI reduced the number of items, but information about other apartments was missing. 50% of the participants noted that increasing the search range of FUI adds items to the screen, causing more clutter and making comparisons more difficult.

In FUI, the visual clutter and the lack of detail compared to the glyph representation made finding and comparing items difficult. We speculated that this can be remedied by adding a similar glyph representation as in HUI to FUI. Therefore, we performed a follow-up study, which included a condition similar to FUI, but this time using the same glyph representation than HUI.

8.2 Different Degrees of Clustering

We conducted a comparative evaluation of three variations of POI clustering to assess if the amount of shown information affects the performance in a search and in a successive recall tasks.

We investigated three clustering conditions in a between-subjects design. In the first condition, there was no clustering, and all the leaf nodes were displayed at the same time to the user (**LUI**) (see Figure 10(b)). Note that *LUI essentially corresponds to FUI* of the previous study, but now using the same glyph representation as HUI to make it easier to compare different items. Therefore, LUI corresponds to a condition resembling a common AR browser, which uses a filter interface. However, in this study, we reason that the filter parameters lead to a large amount of selected data, thereby, filling the whole screen.

In the second condition, a clustering algorithm was introduced, in which items were grouped by proximity and the parameters set by the user (**SUI**) (see Figure 10(c)). In this condition, the user was able to unfold groups of items. Once the items were revealed, the leaf nodes were not merged back into clusters automatically. Therefore, after a certain number of interactions, the display was populated by a growing number of leaf nodes, similarly to LUI. Finally, the third condition was **HUI**, as used in the previous study (see Figure 10(a)). In both HUI and SUI, participants could merge back items into clusters manually by sequentially undoing their unfold operations. Additionally, HUI also performed automatic decluttering according to the described algorithm.

Hypothesis. LUI showed all available items, independent of their relevance to the user. After a number of interactions in SUI, a similar situation occurs, because the unfolded leaf nodes remain visible on the screen. In HUI, the unfolded leaf nodes are regrouped into clusters, after additional groups are unfolded. We expected the leaf nodes populating the display in LUI and SUI to produce visual clutter and to affect the performance in a search and selection task and in a successive recall task of the previously selected items.

Scenario and Setup. We reused the apartment search scenario of the first study, but conducted the second study at a different location. The interface had the same functionality as the HUI interface in the previous study. We added functionality to perform the recall task. By pressing a button, the superimposed glyphs were removed, and only the view through the camera was shown. A participant

(a) (b) (c)

Figure 10: The interfaces used in the second study differed in the way items were clustered. (a) Adaptive information density display (HUI) with hierarchical clustering, after unfolding a number of groups. (b) Interface without clustering (LUI) showing all data at once. (c) Interface with simple clustering (SUI) based on spatial proximity and similarity.

could indicate the location of a previously selected item by tapping on the screen to mark its location with a white square. The apartment attributes remained the same, except for the text of the leaf node, which indicated the size of the apartment in square meters.

Task. Participants were asked to perform two tasks. The first consisted of a search and selection task, in which participants had to find and select all the apartments matching the characteristics indicated by the experimenter. In the second task, they were shown the surroundings without any digital information superimposed, and they were asked to indicate the locations of the apartments that they remembered from the search and selection task.

Procedure. On the day of the test, participants were first briefed on the experimental procedure and aims. Then they gave informed consent to take part in the study. After they had filled in a brief questionnaire collecting background information, they were led to the spot where the test took place. First, participants were instructed how to operate the interface. Then, they were allowed to practice with the interface, until they felt confident. Next, the experimenter asked participants to search and select all the apartments matching certain characteristics: a size from 80 to 90 m^2, in the highest price category and with the largest number of rooms. In total, there were 20 items matching the required characteristics. Participants were told to alert the experimenter, when they believed they had found all the items. Participants were instructed to be careful in performing the task, as they would be required to carry out a second task based on the first one. There were not explicitly told about the recall task, in order to prevent the use of mnemonic strategies. When participants told the experimenter that they had concluded the search and selection task, they started the recall task. Again, participant were asked to tell the experimenter when the task was completed. Finally, we asked the participants to complete a brief questionnaire to collect their opinions and impressions.

Participants. 36 participants (18 female) volunteered in the study, 12 in each condition. The mean age of was 24.27 (sd=2.5). Subsamples were composed of 50% women and balanced for age, as confirmed by a one-way ANOVA, $F(2)$=.067, p=.93. All participants had very limited experience with AR, if any.

Results. The number of correctly selected items, the number of wrongly selected items and the task durations were compared across the three conditions with a one-way ANOVA (see Figure 11). Statistical analysis revealed no significant differences in the number of correct selections ($F(2)$=1.88, p=.16). Similarly, the time required to complete the task did not significantly differ in the three conditions ($F(2)$=1.54, p=.22). A significant difference emerged in the number of errors users made, when they included apartements into their selection that did not match the requested criteria ($F(2)$=5.04, p=.012). Post-hoc comparison with Bonferroni correction confirmed that with HUI, users made significantly fewer wrong selections ($median$=18.66, sd = 7.04) compared to SUI ($median$=30.18, sd=9.22). A reduction in the number of errors is

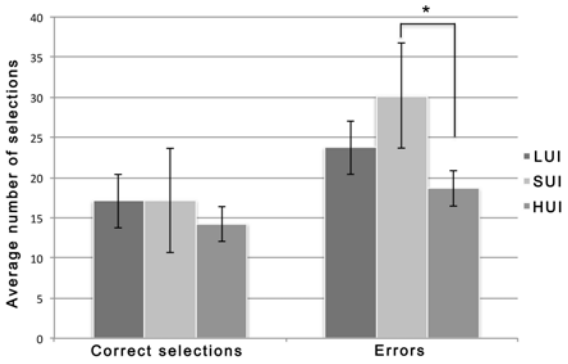

Figure 11: Correct and wrong selections of the search and selection task. When using the adaptive display, participants made significantly fewer wrong selections during the search task.

evident, when comparing the number of wrong selections in LUI ($median$=23.75, sd=9.6) and HUI ($median$=18.66, sd=7.04), even though the difference was not significant.

For the recall task, we computed the relative number of correct recalls as the ratio of the correctly retrieved locations and the number of correct selections of the selection task. Surprisingly, no significant difference emerged comparing the three indexes in a one-way ANOVA ($F(2)$=.26, p=.76).

Regarding the post-use questionnaires, a Kruskal-Wallis test showed no significant differences in the way users evaluated the three systems in terms of information organization, ease of use, involvement, pleasantness of use and satisfaction. A one sample t-test run against the central value of the response scale, i.e., 3, showed a trend ($t(11)$=2.22, p=.04) towards preferring HUI ($median$=3.6, sd=.9) in contrast to LUI ($median$=3, sd=.9) and SUI ($median$=3.4, sd=.79).

Discussion. The significant difference in the error rate during the search and selection task indicates that participants were more focused on selecting the relevant items in the HUI condition. The adaptive interface of the HUI condition actively avoids showing results that do not correspond to the current selection of the user, thereby reducing visual clutter and decreasing the chance of distracting the user from the currently relevant items.

Interestingly, there was no significant difference in identifying the found items in the recall task. We believe that the reason for this is that the amount of clutter was not high enough to impact the perception of the presented data. To allow for a fair comparison between the three interface conditions, we limited the number of items in the surroundings to a small amount (219 items). The number was chosen in a way so that LUI could still arrange the items on

the screen and the items were still clearly visible and distinguishable. However, this generally reduced the amount of clutter, and participants did not seem to have issues identifying items in the LUI and SUI interface, even though the screen was covered with items. Nevertheless, the post-use questionnaires revealed a trend towards participants being more satisfied with the adaptive interface (HUI).

9 Conclusion

We presented a method that reduces the visual clutter created by placing items in the view of an AR browser. Our method balances the amount of presented information against the potential clutter created by placing items on the screen. This information density display for AR uses hierarchical clustering to create a data structure that allows us to select the appropriate level-of-detail for the available display space. Our method selects single items and groups of items from the hierarchy based on user-defined preferences and the amount of visual clutter.

We conducted a qualitative pilot study to compare our interface to a conventional AR browser interface. Users clearly preferred our interface, because it provided a better overview of the data and allowed for easier comparison. We conducted a second study to evaluate the effect of different degrees of clustering. Users generally made fewer errors when using our interface for a searching, which indicates that the reduced clutter allowed them to focus on finding the relevant items. There was a trend towards preferring our interface. However, the preference was not as clear as in the pilot.

We will perform a follow-up study with a larger number of items to identify additional differences when users are exposed to different degrees of clustering. We believe that clutter may have a stronger influence on search and recall tasks when users must relate the items to the real world context. Users of AR applications generally must relate the augmented display space shown on the display device to features of the real world context recorded by the video camera. With an increasing amount of clutter, the augmented display space becomes increasingly occluded. Therefore, it may be more difficult to identify features of the world that items relate to.

In addition, we will investigate methods to take real world geometry into account during the hierarchical clustering phase in order to avoid grouping items that are located, e.g., in different floors of a building. We will furthermore improve the interface to incorporate additional location specific criteria for setting user preferences.

Although we did not test it, we speculate that mixing 2D content (e.g., conventional heads-up displays or menus) and 3D content (POI labels) can easily be integrated into our system. If the application places 2D content independently of 3D content, the associated regions can be simply marked as occupied in the layout algorithm and will not be considered for POI placement. If 2D content is related to 3D content, it can be treated as a special class of label. Assuming that the 2D content fits into a rectangular canvas, the layout optimization has to be extended to consider the size of the labels and not just the number of labels. The overall optimization approach need not be changed.

While the presented interface is targeted at mobile phones, the current state-of-the-art AR platform for the consumer market, the algorithm can also be deployed on head-mounted displays to reduce visual clutter. Avoiding visual clutter for head-mounted displays is even more critical than for mobile phone displays. When using a mobile phone, a user has the AR view of the phone and an unobstructed view of the real world. However, head-mounted displays overlay information directly in the field-of-view of the user, thereby obstructing the only available view. Hence, the information density must be controlled so that head-mounted displays can provide convenient access to location-based information without blocking the view. We believe that when knowledge about the world becomes widely available adaptive information density displays such as the one presented in this paper will become indispensable.

Acknowledgements

This work was partially funded by the European Union (FP7-ICT-601139 "CultAR", FP7-ICT-611526 "Magellan"), by the Austrian Science Fund (FWF) under contract P-2402 and by the Christian Doppler Society ("CDL Handheld Augmented Reality").

References

[1] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. Recent advances in augmented reality. *Computer Graphics and Applications, IEEE*, 21(6):34 –47, Dec. 2001.

[2] M. Balzer and O. Deussen. Level-of-detail visualization of clustered graph layouts. In *APVIS '07.*, pages 133–140, Feb 2007.

[3] B. Bell, S. Feiner, and T. Höllerer. Orlando, Florida.

[4] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93*, pages 73–80, 1993.

[5] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[6] A. Dix and G. Ellis. by chance: enhancing interaction with large data sets through statistical sampling. In *AVI '02*, pages 167–176, New York, NY, USA, 2002. ACM.

[7] G. Dueck and T. Scheuer. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1):161 – 175, 1990.

[8] N. Elmqvist and J.-D. D. Fekete. Hierarchical aggregation for information visualization: overview, techniques, and design guidelines. *IEEE TVCG*, 16(3):439–54, June 2010.

[9] S. Feiner, B. Macintyre, and D. Seligmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–62, July 1993.

[10] G. W. Furnas. Generalized fisheye views. *CHI '86*, 17(4):16–23, Apr. 1986.

[11] D. Holten. Hierarchical edge bundles: visualization of adjacency relations in hierarchical data. *IEEE TVCG*, 12(5):741–8, Jan. 2006.

[12] S. Julier, Y. Baillot, D. Brown, and M. Lanzagorta. Information filtering for mobile augmented reality. *IEEE CG & A*, 22(5):12–15, 2002.

[13] S. Maass and J. Döllner. Dynamic Annotation of Interactive Environments using Object-Integrated Billboards. In *WSCG'06*, pages 327–334, 2006.

[14] C. D. Manning, R. Prabhakar, and H. Schütze. *Introduction to Information Retrieval*. Cambrige University Press, 2009.

[15] E. Mendez, D. Kalkofen, D. Schmalstieg, and E. Méndez. Interactive context-driven visualization tools for augmented reality. In *ISMAR '06*, pages 209–218, 2006.

[16] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., 1984.

[17] M. Tatzgern, D. Kalkofen, R. Grasset, and D. Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3d space. In *IEEE VR'14*, March 2014.

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

[19] M. Tatzgern, D. Kalkofen, and D. Schmalstieg. Dynamic compact visualizations for augmented reality. In *IEEE Virtual Reality (VR)*, pages 3–6, Mar. 2013.

[20] M. Tatzgern, D. Kalkofen, D. Schmalstieg, and Schmalstieg, Dieter. Compact explosion diagrams. In *Computers & Graphics*, volume 35, pages 135–147, New York, New York, USA, June 2010. ACM Press.

[21] K. Virrantaus, J. Markkula, A. Garmash, V. Terziyan, J. Veijalainen, A. Katanosov, and H. Tirri. Developing GIS-supported location-based services. In *WISE'01*, volume 2, pages 66–75, 2001.

[22] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. Real-time panoramic mapping and tracking on mobile phones. In *Proc. IEEE Virtual Reality*, pages 211–218, Boston, USA, March 2010.

[23] M. O. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3/4):194–210, Dec. 2002.

[24] A. Woodruff, J. Landay, and M. Stonebraker. Constant density visualizations of non-uniform distributions of data. UIST '98, pages 19–28, New York, NY, USA, 1998. ACM.