A Partially-Sorted Concentric Layout for Efficient Label Localization in Augmented Reality

Zijing Zhou, Lili Wang, and Voicu Popescu

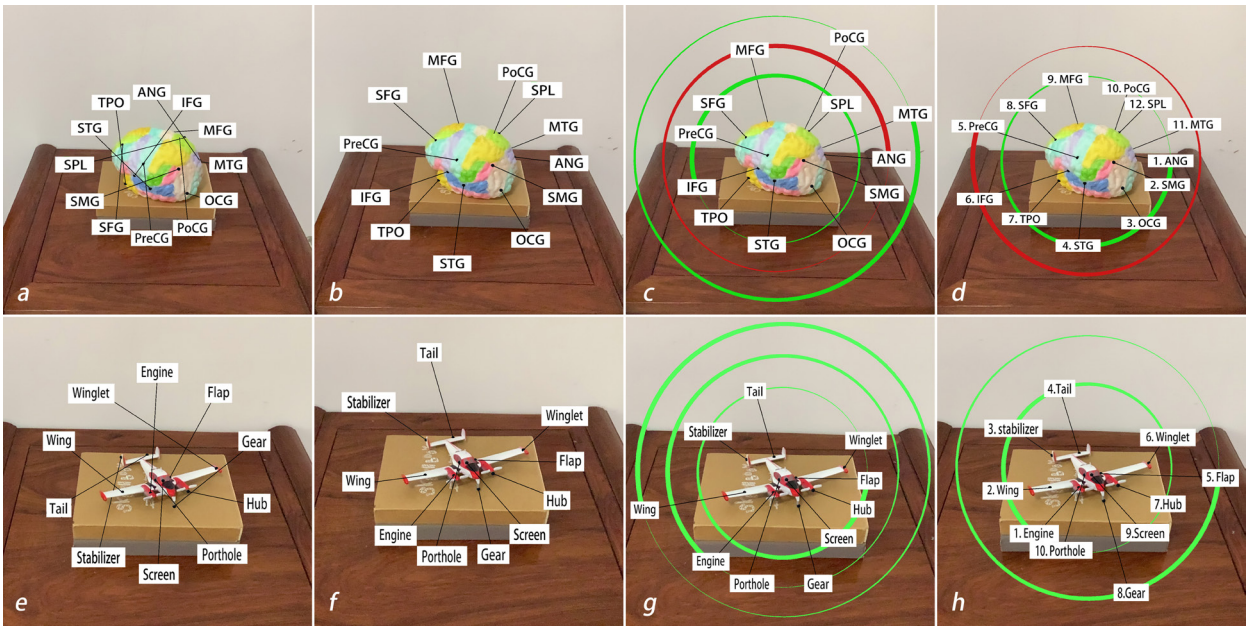

Fig. 1: Real world brain (top) and airplane (bottom) models labeled by an AR application. Finding a label can be accelerated by alphabetical sorting, but simply sorting all labels results in leader line intersections (*a* and *e*). A single-circle layout that avoids intersections cannot sort the labels (*b* and *f*). Our method computes a concentric label layout that avoids intersections while sorting the labels on each circle (*c* and *g*). An optional label presorting step reduces the number of circles to two (*d* and *h*).

Abstract—A common approach for Augmented Reality labeling is to display the label text on a flag planted into the real world element at a 3D anchor point. When there are more than just a few labels, the efficiency of the interface decreases as the user has to search for a given label sequentially. The search can be accelerated by sorting the labels alphabetically, but sorting all labels results in long and intersecting leader lines from the anchor points to the labels. This paper proposes a partially-sorted concentric label layout that leverages the search efficiency of sorting while avoiding the label display problems of long or intersecting leader lines. The labels are partitioned into a small number of sorted sequences displayed on circles of increasing radii. Since the labels on a circle are sorted, the user can quickly search each circle. A tight upper bound derived from circular permutation theory limits the number of circles and thereby the complexity of the label layout. For example, 12 labels require at most three circles. When the application allows it, the labels are presorted to further reduce the number of circles in the layout. The layout was tested in a user study where it significantly reduced the label searching time compared to a conventional single-circle layout.

Index Terms—Augmented Reality, Label layout, Fast label finding

1 Introduction

Augmented Reality (AR) is a powerful interface that allows overlaying graphical annotations directly onto the real world scene. The annotations appear into the field of view of the user, and, as the user changes their view, the annotations remain anchored to the real world elements that they describe. A popular type of AR annotation is a text label displayed on a virtual flag, with the flagpole acting as a leader line, anchored to the real world element described by the label. For example, AR labels can name the parts of a complex mechanical assembly, or of an anatomical physical model. Displaying the label using a flag provides sufficient screen real estate for the label text. It avoids that the label occludes the element it describes, and it indicates the element accurately and unambiguously. As the user changes view, the labels are continually rearranged to keep them close to the element they describe, to avoid leader line intersections and collisions between labels.

Consider the scenario of an anatomy class where each student has their own anatomical model to follow along the description of the instructor. As the instructor mentions various anatomy elements, each student has to quickly locate the anatomy element on their model. An AR application should help students do so even without the knowledge

- *Lili Wang is with State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, Beijing, China; Peng Cheng Laboratory, Shengzhen, China; and Beijing Advanced Innovation Center for Biomedical Engineering, Beihang University, Beijing, China. Lili Wang is the corresponding author. E-mail: wanglily@buaa.edu.cn.*
- *Zijing Zhou is with State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, Beijing, China. E-mail: 418845530@qq.com. Voicu Popescu is with Purdue University, U.S. E-mail: popescu@purdue.edu.*

Manuscript received 15 Mar. 2021; revised 11 June 2021; accepted 2 July 2021. Date of publication 27 Aug. 2021; date of current version 1 Oct. 2021. Digital Object Identifier no. 10.1109/TVCG.2021.3106492

of the order in which the instructor is going to mention the various elements. Similarly, when a technician is called upon to service an old mechanical assembly based on a textual description they hold in their left hand and with the help of an AR application deployed on a tablet they hold in their right hand, the technician should be able to locate the parts mentioned in the text quickly. Locating a specific label can be time consuming if the user has to examine the labels sequentially, one at the time. Using conventional search mechanisms developed for keyboard-based human-computer interfaces, which trigger searching by pressing specific key combinations, introduces unwanted focus shifts, which diminish the fundamental benefit of an AR interface.

In this paper we present a method for displaying AR labels in a way that allows the user to locate a specific label efficiently. Our intuition is that if the labels are sorted alphabetically, the user can locate a specific label efficiently by skipping over some of the other labels and converging more quickly on the label of interest. However, simply sorting all the labels is not feasible as it violates the important label display constraints of proximity to the element described, of avoiding collisions between labels, and of avoiding intersections between leader lines. Instead, our method computes a partial sorting of the set of labels by partitioning the set into a small number of sorted sequences that respect the label display constraints. The labels are displayed on concentric circles surrounding the real world entity, with one sorted sequence per circle, which allows the user to search each circle efficiently.

In Fig. 1, simply sorting all labels (*a* and *e*) yields leader line intersections, as the sorting of the anchor points around the center of the brain and airplane models does not match the alphabetical sorting of the labels. A single circle layout that avoids intersections has to give up on sorting the labels (*b* and *f*). Our method partitions the set of labels into a few sorted sequences, each displayed on a circle (*c, d, g, h*). The circle line thickness decreases from the first to the last label in the sequence; this allows the user to locate quickly the first label of the sequence, e.g., "IFG" for the smallest circle in image c, from where to start the search on the circle. The labels on a circle are sorted either clockwise (green) or counterclockwise (red), as having the two options allows for longer sequences and fewer circles. Our method works either with a given label ordering, e.g., the alphabetical order of the label text strings, or with a custom ordering of the label sequences that it precomputes to maximize sequence length and minimize the number of circles. In Fig. 1 the optimized numbering reduces the number of circles from three to two (*c* and *g* versus *d* and *h*).

Our method leverages the advantage of sorted labels while keeping the labels close to the real world element they describe, and while avoiding leader line intersections. For our method to be effective, the number of circles in the layout should be as small as possible. A recent mathematics research result that extends the classical Erdős–Szekeres theorem from linear to circular permutations [17] provides a tight upper bound on the number of circles needed by our layout. For example, three circles are always sufficient for 12 labels, four circles are always sufficient for 18 labels, and five circles are always sufficient for 24 labels. Whereas these are upper bounds, the actual number of circles is often smaller, e.g., two circles for the 12 labels in Fig. 1, *d*.

We have tested our partially-sorted concentric layout for AR label display in a user study where ($N = 34$) participants had to locate labels as quickly as possible. The study had three conditions: the conventional approach of displaying the labels unsorted on a single circle, as the control condition (CC), our method *without* the optional presorting, as a first experimental condition (EC1), and our method *with* the optional presorting, as a second experimental condition (EC2). The study shows that for both variants of our method (EC1 and EC2) participants found labels significantly faster than for the single-circle condition (CC). Furthermore, the optional presorting reduced the label finding time significantly for our method (EC2) over not using the presorting (EC1). Both experimental conditions had significantly lower task load scores (RTLX [24]) than the control condition, as well as significantly higher scores in our usability questionnaire.

In summary, the contributions of our paper are as follows: (1) a partially-sorted AR external label layout with bounded complexity; (2) an optional label pre-sorting step that further reduces the average layout complexity; (3) a user study ($N = 34$) where our layout has been shown to allow finding AR labels faster than a conventional one-circle layout.

2 Previous Work

Label layout is a challenging problem. A frequent use case is for medical visualizations, as inventoried by Oeltze-Jafra and Preim [37]. Marks et al. [35] investigated the computational complexity of cartographic label placement, i.e., the problem of laying out labels to describe points of interest on a 2D map. They have shown that finding an admissible label placement, i.e., finding a layout that avoids that labels overlap with other labels or with points of interest, and that places the labels within a given distance of their point of interest is NP-complete. Furthermore, they have shown that finding an optimal label layout, which minimizes overlap and distance from label to point of interest can only be done in polynomial time if and only if $P = NP$. Consequently, in the context of real-time label placement, as that of an interactive AR application, optimal label placements cannot be guaranteed and the goal is to quickly achieve a suitable placement.

Existing label layout methods can be classified into two main categories according to the where the labels are placed. Internal label layout methods place the annotations directly on the object they describe (Sect. 2.1). External label layout methods place the annotations around the object, connecting the annotations to anchor points through leader lines (Sect. 2.2). Hybrid methods use both internal and external labels (Sect. 2.3). We also review prior work on guiding the user's attention (Sect. 2.1), and we end the discussion of prior work with an overview of the approaches for evaluating label layouts (Sect. 2.5).

2.1 Internal label layouts

Ropinski et al. [41] proposed an internal labeling method that shapes and orients the label according to the shape of the feature it describes. Kouřil et al. [30] partitioned the scene hierarchically into several levels of detail, each with its own set of labels, and displayed the labels of the level of detail corresponding to the current user view zoom level. Han et al. [23] proposed a method that only labels the features inside a user's circular lens, and, when collisions occur, the method switches from an internal to an external label layout that simply highlights the relevant labels in a list displayed at the periphery of the image. In interactive 3D graphics applications, such as AR, the alignment of the user view with the scene changes, so internal label layouts that are constrained by the scene geometry often result in ambiguity, and external label layouts should be preferred [15].

2.2 External label layouts

Compared with internal labels, external labels have more applications in complex scenes. For example, Balata et al. [4] proposed using external tags to improve the efficiency and accuracy of selecting within a dense set of small objects. For a comprehensive review of external label layout methods we refer the reader to a recent survey [9].

One approach for external label placement is the *boundary label layout*, which defines an area in the user view and places the labels on the boundary of the area. Bekos et al. [8] fixed the position of the labels to the right of a rectangular area, and anchor points are adjusted within the feature to which they are attached to relax the leader line routes; the goal is to minimize the total length of the leader lines while ensuring that the leader lines do not intersect. In subsequent work [7], they studied boundary labeling with octilinear leader lines, i.e., piecewise linear lines made from horizontal, vertical, and diagonal segments. More recently, Bekos et al. extended their work to scenarios where multiple anchor points are connected to the same label [6]. Gemsa et al. [20] proposed a boundary labeling method applied to panoramas, in which the labels are laid out in multiple lines on the upper border that typically corresponds to a less interesting part of the scene, such as the sky. Kindermann et al. [29] avoid the multiple lines by placing labels both on the upper and lateral boundaries. Boundary labeling is more suitable for two-dimensional scenes such as maps. For three-dimensional augmented reality scenes, placing the label around the object is a better option for label layout.

Another approach, suitable for outside-looking-in visualization scenarios where a single object is inspected with a view that encompasses it, is to use the silhouette line of the object to place the labels. Hartmann et al. [26] proposed a method for arranging labels at the periphery of an object using forces defined by a dynamic potential field. In subsequent work [2], they proposed different label arrangement methods, including based on a spring system, a computational infrastructure that our works also relies upon. Shibata et al. [44] proposed label layout as a view management approach that would tweak the position of the labels to improve the current view in terms of reducing overlap and intersections, an approach that was sufficiently flexible to adapt to the low computational power of early generation mobile devices.

The typical greedy solution to the NP-hard problem of label placement does not pay special attention to the order of placement. Therefore, the labels placed early occupy the best positions, and it becomes increasingly more difficult to place subsequent labels. Stein et al. [45] proposed an improved greedy algorithm that relies on a heuristic to find the labels that are more difficult to place and to place them early, which leads to better approximate solutions.

Grasset et al. [22] proposed an image-based method that recognizes important areas in the image to automatically generate constraints for labels. Tatzgern et al. [48] create compact label layouts for AR by avoiding redundant labels when multiple instances of the same feature are visible. The method is prone to intersection of leader lines when the viewpoint moves. In subsequent work [47], they proposed several geometric constraints to optimize the label position, avoiding leader line intersections. Later they improved readability through clustering to reduce the density of information on the screen [49]. The method of Orlosky et al. [38] detects important image regions which should not be obscured by the labels, such as the faces and hands of computer animation characters. Čmolík et al. [13] optimizes the labeling problem on models with perspective effects. This method takes into account the visual overlap and opacity of objects.

2.3 Hybrid internal/external label layout

Researchers have also attempted to leverage the benefits and avoid the challenges of the internal and external label layouts by proposing hybrid label placements. Bell et al. [10] proposed a label layout optimization method for AR that places labels internally for as long as possible and reverts to an external placement for labels that cannot be placed internally. Götzelmann et al. [21] proposed *annotation boxes* as a third type of label, in addition to internal and external labels; annotation boxes do not have leader lines and are anchored solely by proximity, which simplifies constraints. Löffler et al. [33] propose polynomial algorithms that maximize the number of internal labels in hybrid layouts. Čmolík et al. [14] have recently proposed a real-time mixed labeling algorithm that provides the application the flexibility of tuning the partitioning of labels between the internal and external subsets on a continuum. The algorithm avoids overlap between internal labels. The issue of external label leader line intersection is bypassed by requiring all lines be horizontal. The layouts are evaluated with the help of professional illustrators, and label finding times are not a focus.

2.4 Approaches for guiding the user's attention

Label layout approaches can also be classified based on the approach taken to guide the user's attention towards the label of interest.

One approach is to rely on color coding, which brings an additional dimension for sorting labels. Fekete et al. reduces the ambiguity of dense labels by using label box borders that match the color of the part of the object to which they correspond [19]. The use of color to make more salient the information conveyed by images is of course part of the classical visualization toolbox [11], with color aiding to convey both quantitative and qualitative aspects of the data shown to the user. In AR, the use of complex color palettes is difficult when true opacity is not supported, e.g., for see-through AR head-mounted displays (HMDs). We rely on two colors, red and green, to convey the order of the sorting of the labels on the circles of our layouts.

Another approach is to rely on gaze tracking to reduce the complexity of selection in a 3D environment, by catering dynamically to the current user's view direction. The approach was used in VR [12], and more recently in the context of AR games [31, 39]. The approach requires eye tracking, which is not yet a standard feature of AR displays, and it is useful when the user has a general idea of where to look for the object/label of interest, without which the user is still required to perform a sequential search of the scene. Eye tracking is particularly useful in the context of a large number of labels, e.g. in the inside-looking-out visualization scenario, where eye tracking helps first select the object of interest. As such, eye tracking is complementary to our work, which focuses on the outside-looking-in scenario, where it speeds up the searching in the set of labels of an object of interest.

A third approach is to filter labels intelligently based on the current user's actions, such as reducing the labels to those pertinent to the current level of detail [30]. This approach is also orthogonal and complementary to our approach. Additional approaches such as using multiple–juxtaposed or coordinated–views are possible, but these information visualization approaches are less suitable for AR interfaces where the spatial layout of the real world scene is both hard to change and important to convey as is, and where screen real estate is limited.

We take the approach of using leader lines to guide the user's attention from the anchor point to the label text and vice versa, an approach long used in AR [3]. Our contribution is to resolve the competing interest between label sorting and anchor point location, achieving a partial sorting that accelerates label finding.

2.5 Evaluation of label layout

With the proliferation of label layout approaches also came a series of studies that do not propose new label layouts bur rather focus on comparative evaluations of various prior art layout approaches, for various metrics, and in various scenarios. Azuma et al. [3] evaluated four label layout styles under augmented reality in terms of algorithm efficiency, label overlap count, and user response time. Hartmann et al. [27] classified the label layout styles and proposed several metrics for label functional requirements and aesthetics. Madsen et al. [34] conducted user experiments that compared the difference in terms of view update frequency, between object space and image space label rendering. Li et al. [32] tested users' subjective preferences for label layout styles generated according to different penalty functions. Barth et al. [5] studied the readability and aesthetics of different types of guide lines in boundary labeling.

Our method builds on prior work label placement efforts. Our method opts for an external label placement, which has been shown to be most suitable for an interactive AR application. We target the outside-looking-in scenario where a complex 3D entity is seen in its entirety from various external viewpoints surrounding the 3D entity. Our method leverages partial sorting to accelerate finding a given label. Our layouts are not optimal, but we provide a tight upper bound on the complexity of the layout, in terms of the number of circles. Our user study focuses on the objective metric of the time needed to a user to locate a given label, as well as on subjective metrics such as self reported task load and usability.

3 Partially-Sorted Concentric Layout

Our method assigns labels to concentric circles, making sure that the labels of a same circle are sorted, which allows the user to quickly check whether a given label is on a circle or not. For our partially-sorted concentric label layout to be effective the number of circles has to be as small as possible. Recently the Erdős–Szekeres theorem has been extended from linear to cyclic permutations [17], which provides an upper bound on the number of circles in our layout, as described in Section 3.1. The remaining subsections give our overall iterative layout computation algorithm (Section 3.2), the algorithm used to compute a candidate label layout at each iteration (Section 3.3), the strategy for obtaining layout temporal coherence (Section 3.4), and an optional presorting of labels that further improves layout efficiency (Section 3.5).

3.1 Upper bound on the number of circles

The classical Erdős–Szekeres theorem [18] establishes that any sequence of length $kl+1$ has an increasing sorted subsequence of length

$k+1$ or a decreasing sorted subsequence of length $l+1$. The guarantee of the presence of such a monotonic sequence is useful in deriving a lower bound on the number of labels assigned to one circle, and, consequently, in deriving an upper bound on the total number of circles. Recently, the Erdős–Szekeres theorem has been extended to circular sequences, by showing that a cyclic permutation of length $kl+2$ has either an increasing cyclic sub-permutation of length $k+2$, or a decreasing cyclic sub-permutation of length $l+2$ [17]. Informally, a cyclic sub-permutation is a permutation that can wrap around the original sequence by jumping from the last to the first or the first to the last element. This extension is useful in our context since the circles of our layout can accommodate cyclic sorted sequences, and not just conventional (linear) sorted sequences, relaxation that allows for finding longer sorted subsequences for an overall smaller number of circles. For example, for the sequence 7, 2, 4, 5, 3, the longest monotonic linear subsequence is 2, 4, 5, whereas the longest monotonic cyclic subsequence is 2, 4, 5, 7.

Using the extended Erdős–Szekeres theorem, given $kl+2$ labels, there exists a sorted cyclic subsequence of length $\max(k+2, l+2)$. The worst case, i.e., the shortest such sorted cyclic subsequence, occurs when $k=l$ (Equation 1, where s is the number of labels).

$$s = kl+2 = k^2+2 \tag{1}$$

Consequently,

$$k = \lfloor \sqrt{s-2} \rfloor \tag{2}$$

for a shortest sorted cyclic subsequence of length l_{min} of

$$l_{min} = \lfloor \sqrt{s-2} \rfloor + 2 \tag{3}$$

Consequently, an iterative construction of the layout places at least l_{min} labels at each iteration. The number of layout circles can be found as the number of iterations needed to place all labels. Equation 4 gives the recursion that models the reduction in the number of labels s achieved by each iteration. The termination condition is $s \leq 1$.

$$s = s - \lfloor \sqrt{s-2} \rfloor - 2 \tag{4}$$

Table 1 shows the maximum number of circles (row 2) needed to accommodate s labels (row 1). Two circles are always sufficient to accommodate 7 labels, and three circles are always sufficient for 12 labels. Whereas row 2 gives the maximum number of circles needed, i.e., in the worst case scenario, row 3 gives the average number of circles needed, computed over all possible permutations for $s \leq 10$, and over ten million random permutations for $s > 10$. Row 3 confirms that the expected number of circles is lower than the upper bound given by row 2, e.g., 13 labels are likely to require only three circles, and not four, and 25 labels are likely to require only four circles, and not six.

Table 1: Number of layout circles for a given number of labels.

Labels	3	4	7	8	12	13	18	19	24	25
Max. circles	1	2	2	3	3	4	4	5	5	6
Avg. circles	1	1.4	1.9	2.0	2.9	3.0	3.6	3.8	4.1	4.2

3.2 Label Layout Algorithm

We compute label layouts with an iterative process. At each iteration, our algorithm creates a candidate label layout. The candidate layout is constructed by sorting radii emanating from the center of the object and ending at individual labels. The concurrency of these radii guarantees that they do not intersect. These radii are used as helpers to define leader lines. Since leader lines originate from different anchor points, the non-intersection is not guaranteed. When such an intersection occurs, corrective forces are computed and used in greedy fashion in the subsequent iteration to eventually remove all intersections. We have strong empirical evidence that the algorithm converges, i.e., it always converged in fewer than 18 iterations, and on average in 5.4 iterations in our experiments with multiple scenes and with up to 16 labels.

Consider a 3D entity (e.g., an anatomical model, a mechanical assembly) that is examined interactively by a user of an augmented reality application from viewpoints surrounding the entity. Given the entity's 3D geometry G and a set L of n labels , we compute for each user view V a partially-sorted concentric label layout with Algorithm 1.

Algorithm 1 Partially-Sorted Concentric Label Layout

Input: 3D entity G, set of labels L, and user view V
Output: label layout $L.\{r,d\}$, layout plane Π, layout center C
1: $\Pi \leftarrow$ Plane($V.vd$, $C \leftarrow$ Centroid(G))
2: $r_0 \leftarrow$ Circumscribe(G, V, Π)
3: **for** $L_i \in L$ **do** $P_i \leftarrow$ Project(A_i, V, Π)
4: $F \leftarrow \varnothing$; $iter = 0$;
5: **repeat**
6: $\quad L.\{r,d\} \leftarrow$ CandidateLabelLayout($L.\{T,P\}, C, r_0, F$)
7: $\quad I \leftarrow$ IntersectionDetection($L.\{r,d\}$)
8: $\quad F \leftarrow$ Adjustment(I)
9: $\quad iter$++
10: **until** $I == \varnothing \vee iter ==$ MAXITERATIONS
11: **return** $(L.\{r,d\}, \Pi, C)$

In Algorithm1, each label L_i has associated a text string T, a 3D anchor point A, a radius r of its layout circle, a direction d from the object center to the label, a projection P of its anchor point, and a flag a which indicates whether the label has already been placed on a circle. T and A are provided as input for each label, and the layout is set by computing r and d for each label.

The labels are laid out in a plane Π that passes through the centroid C of the 3D entity G and has a normal aligned with the user view direction $V.vd$, i.e., plane Π is parallel to the image plane (line 1). The smallest label circle with radius r_0 circumscribes the projection of 3D entity G onto plane Π from the user view V (line 2). The label anchors A_i are projected onto plane Π from V to P_i (line 3).

At each iteration $iter$, a candidate layout is computed by setting the r and d fields of each label (line 6). The candidate label layout algorithm is given in Algorithm 2. The candidate layout is examined for leader line intersections (line 7), and a set of corrective forces F is computed based on the intersections (line 8). A pair of corrective forces is defined for each pair of labels with intersecting leader lines. The two forces of the pair are tangent to the circle of the layout, with directions that pull the labels towards each other, aiming for the labels to switch place to undo the leader line intersection. The forces have equal magnitude, which increases from iteration to iteration until the leader line intersection is removed. The set of corrective forces is used at the next iteration to update the layout (line 6).

Fig. 2 illustrates our algorithm on a set of labels (left); the first iteration results in leader line intersections for labels H and D (middle), which are addressed with a set of corrective forces, resulting in the final two-circle partially-sorted concentric layout (right). A pair of corrective forces is defined for each pair of labels with intersecting leader lines, e.g., F_{DH} and F_{HD} in Fig. 2, middle. The iterative process stops when the updated layout has no intersections, or when the maximum number of iterations (MAXITERATIONS) has been reached (line 10). In practice, we set MAXITERATIONS to 100, but the algorithm always converged in fewer than 20 iterations.

Once the layout is computed, the labels are rendered, each on a circle, with the labels on the same circle being sorted. The position of a label L_i is defined by the intersection between the circle of radius r_i and direction d_i. Clockwise sorted circles are drawn in green, and counterclockwise in red (Fig. 2, right). The circle line thickness is largest for the first label of the circle sequence, and it decreases with subsequent labels, which allows the user to quickly find the starting point for their search on each circle (e.g., labels A and B on the two circles in Fig. 2, right). The leader is drawn from the label position to the label anchor point.

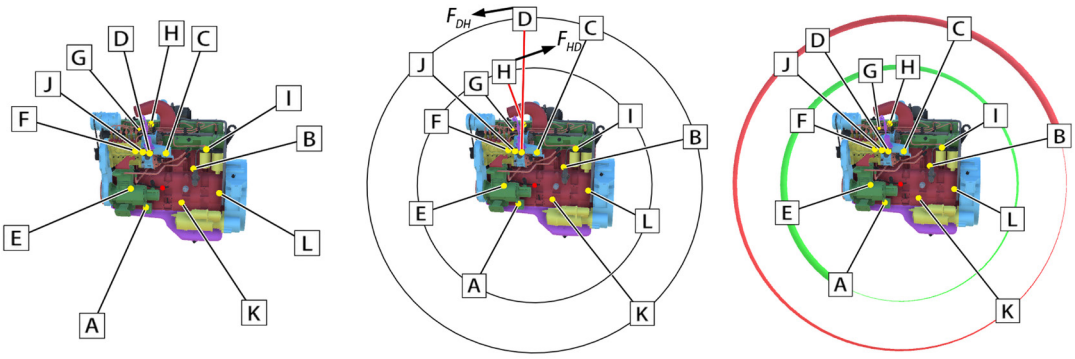

Fig. 2: Label layout computation with Algorithm 1: labels to be placed (left), layout computed by the first iteration with intersection between leader lines of labels D and H and corrective forces F_{DH} and F_{HD} (middle), and final layout computed after the fifth iteration (right).

3.3 Candidate Label Layout Algorithm

Each iteration computes an updated partially-sorted concentric layout according to Algorithm 2, which is illustrated in Fig. 3.

Algorithm 2 Candidate Label Layout

Input: set of labels L, centroid C, first circle radius r_0, and set of corrective forces F
Output: label layout $L.\{r,d\}$
1: **for** $L_i \in L$ **do** $d_i \leftarrow (P_i - C) \,/\, \|P_i - C\|$
2: $L.\{d\} \leftarrow \text{Relax}(L.\{d\}, F)$
3: $L \leftarrow \text{Sort}(L.\{d\})$
4: $r \leftarrow r_0$; **for** $L_i \in L$ **do** $A_i \leftarrow$ false
5: **repeat**
6: $\quad S \leftarrow \text{MaxSortedSubsequence}(L)$
7: $\quad$ **for** $S_i \in S$ **do** $S_i.r \leftarrow r$; $S_i.a \leftarrow$ true
8: $\quad r \leftarrow r + \Delta r$
9: **until** $S == \varnothing$
10: **return** $L.\{r,d\}$

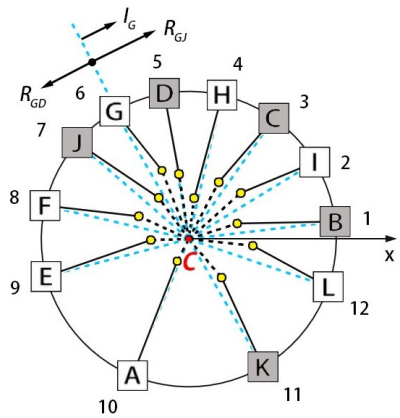

Fig. 3: Candidate label layout Algorithm2. The algorithm finishes in two iterations generating two sorted subsequences "AEFGHIL" and "BCDJK", which will be displayed on two circles.

For each label L_i, its direction d_i is initialized to the direction from the entity center C to the label plane projection of the anchor point P_i (line 1). In Fig. 3 these initial directions are shown with black dotted lines, C is shown with red, and the projected anchor points P_i are shown with yellow. These initial directions are then relaxed to avoid that labels clump together, while striving to keep the label as close as possible to its anchor point and to avoid long leader lines (line 2). In Fig. 3 the relaxed directions are shown with blue dotted lines.

The relaxation process is implemented with a one-dimensional system of springs. The spring system is loaded with three types of forces. One type are repulsive forces R_{ij}, defined between all pairs of labels L_iL_j, with magnitude inversely proportional to the angle $\sphericalangle(d_i,d_j)$, and with direction perpendicular to d_i, and away from d_j. Fig. 3 shows only two of the repulsive forces that act on label G, i.e., R_{GD} and R_{GJ}. The second type are inertial forces I_i, one for each label, with magnitude proportional to the displacement of the label direction from the initial direction CP_i, and with direction towards the initial direction. Fig. 3 shows the inertial force I_G that aims to bring label G back to its initial position, i.e., clockwise, to the right. The third type are the corrective forces F computed at the previous iteration of Algorithm 1, which aim to remove the intersection (see Fig. 2, middle).

Once the directions are relaxed, the labels are sorted based on the angle between their relaxed direction and the x axis (line 3). In Fig. 3, the sorting of the labels based on the relaxed directions (blue dotted lines) is indicated with the number by the label, e.g., 6 for label "G".

Then labels are placed in a concentric layout through an iterative process (lines 4-9). Initially, all labels are marked as not assigned to the layout (line 4), and the process completes when there are no labels left to assign (line 9). At each iteration, the algorithm finds the longest alphabetically sorted subsequence of unassigned labels (line 6). The alphabetical sorting can use the text string T_i of each label L_i, e.g., "Stabilizer" "Tail" "Winglet", a label numbering chosen by the application, e.g., "1" "2" "3", or, when the application permits, a label numbering computed by our method as a preprocess with the goal of maximizing the length of sorted sequences and thereby minimizing the number of circles in the concentric layout (Sect. 3.5).

The longest alphabetically sorted subsequence is found by traversing L, and, for each unassigned label L_i, we find the longest ascending-order subsequence of unassigned labels that starts at L_i, both in clockwise and counterclockwise order. The overall longest subsequence S is assigned to the current circle (line 7). In Fig. 3 the longest subsequence at the first iteration is "AEFGHIL", which is assigned to the first circle, of smallest radius, i.e., r_0. The radius of the circle increases by Δr (line 8), to obtain the concentric layout of labels. The fixed circle to circle radius increment Δr is set based on the label font size, or based on the available space in the image surrounding the entity labeled by the application. Our current implementation does not handle exceedingly long labels, which have to be dealt with by wrapping the text on multiple lines, by adjusting the font size for individual labels, and by modulating the spacing between consecutive circles based on the size of the label boxes. For the example in Fig. 3, the longest sorted subsequence of the second iteration contains all remaining labels, i.e., "BCDJK", which are assigned to the second circle.

3.4 Layout temporal coherence

Algorithm 1 computes the layout based on the user view, and, as the user changes the view, so does the layout. To avoid frequent small changes

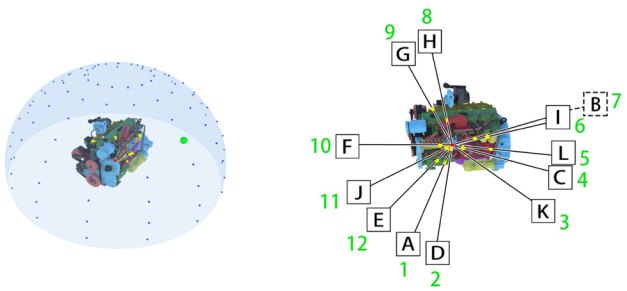

Fig. 4: Left: sampling of possible view directions used in label presorting (Algorithm 3). Right: label presorting iteration corresponding to one of the possible view directions (green dot in top image).

to the layout, the layout is updated only if the user view translates a significant amount (i.e., in practice we use a translation threshold of 30cm), or if the previous layout results in leader line intersections in the current view. When the layout changes, the change is deployed gradually, with the old layout morphing into the new layout with the labels migrating from their old to their new position with a constant velocity (i.e., in practice we use a label morphing velocity of 0.5m/s).

3.5 Optional Label Presorting

When the application permits, the layout can be further improved by presorting the set of labels, which results in a smaller number of circles. For example, labels "Stabilizer", "Tail", and "Winglet" could be presorted by prefixing the label strings with an index number to "1. Tail", "2. Stabilizer", and "3. Winglet". The presorting takes into account the 3D position of the anchor points of the labels and presorts the labels in greedy fashion to reduce the number of layout circles, for a dense sampling of all possible user view directions. Fig. 4, top, shows the possible view directions as points on a bounding hemisphere.

The presorting is done according to Algorithm 3. The algorithm takes as input the set of labels L, each with its 3D anchor point A_i and text string T_i. The algorithm outputs the set of labels with the text strings prefixed with an index number that reflects the presorting.

Algorithm 3 Label Presorting for Improved Layout Efficiency

Input: Set of labels L, view direction sampling resolution $n_{lat} \times n_{long}$, center point C
Output: Presorted set of labels L
1: **for** $L_i L_j \in L \times L$ **do** $M[i][j] \leftarrow 0$
2: **for** $k \leftarrow 0$ to $n_{lat} - 1$ **do**
3: **for** $l \leftarrow 0$ to $n_{long} - 1$ **do**
4: $vd_{kl} \leftarrow$ LatLong2Direction$(k, l, n_{lat}, n_{long}, C)$
5: **for** $L_i \in L$ **do** $P_i \leftarrow$ Project(A_i, vd_{kl}, C)
6: **for** $L_i \in L$ **do** $d_i \leftarrow (P_i - C) \,/\, \|P_i - C\|$
7: $S \leftarrow$ IndexSort$(L, L.d)$
8: **for** $L_i \in L$ **do** $M[i][S[i]]$++
9: **repeat**
10: $t_{ij} \leftarrow$ Max(T)
11: $T_i =$ "$< j >.$ " $+T_i$
12: $M[i, 1...n] = 0; M[1...n, j] = 0$
13: **until** $M == 0$
14: **return** L

The algorithm uses a matrix M_{ij} of size $n \times n$ (where n is the number of labels in the scene) to count how often label L_i appears in position j, over all possible view directions. M is computed by iterating over all possible view directions (lines 2 to 8). For a given view direction k,l, the direction vector vd_{kl} is computed by converting from latitude longitude to a 3D point on the unit sphere centered at C (line 4). The anchor points A_i are projected orthographically on the plane with normal vd_{kl} through C (line 5). The label directions in the projection plane are computed (line 6), and they are sorted based on their angles start from label "A"(line 7). The labels are not actually shuffled, but instead the sorting based on directions is recorded in a permutation S of the label indices. Then, for each label L_i, its frequencies (in row i of M) are incremented based on S. In Fig. 4, bottom, the peripheral green numbers illustrate the permutation S computed for the view direction defined by the green point in Fig. 4, top.

Once the 2D array of frequencies M is known, M is used to presort the labels (lines 10-13). At each iteration, the label with highest frequency is prefixed with the index corresponding to the position where the highest frequency occurs (lines 10-11). Once a highest frequency value is used, it is removed from further consideration by setting its rows and columns in M to 0 (line 12). Each iteration presorts a label, and the process terminates when all labels have been presorted, which is indicated with a frequency table with all zero entries (line 13). In Fig. 1, computing the layout using the prefixed labels reduces the number of circles from three (c and g) to two (d and h).

4 Results and Discussion: User Studies

We have evaluated our layouts first in a pilot study ($N = 10$), to select the most promising parameter configuration, and then in a within-subject controlled user study ($N = 34$), in which we compared the most promising configuration, with and without the optional presorting, to the conventional approach of a single circle layout.

4.1 Pilot study

One configuration parameter pertains to the number of labels on each circle. One option (pilot experimental condition 1, or PEC1) is to start placing labels on the inner most circle and then go towards the outer circles, which results in more labels on the inner circles and fewer labels on the outer circles. A second option (PEC2) is to start from the outer-most circle and go towards the inner circles, resulting in more labels on the outer circles and fewer labels on the inner circles. A third option (PEC3) is to limit the number of labels on each circle, which occasionally leads to needing additional circles.

Another configuration parameter is whether the circle line thickness decrease is used to indicate the first label on each circle, or whether the line thickness is constant. PEC1, PEC2, and PEC3 all use the circle line thickness decrease. A fourth pilot experimental condition PEC4 matches PEC1, i.e., the labels are placed starting from the inner circle, except that the circle line thickness is constant. All four experimental conditions tested in the pilot study used the green and red colors to indicate circles with labels sorted clockwise and counterclockwise.

We recruited $N = 10$ pilot study participants, 5 male, 5 female, of age between 14 and 30 years old. Two of the participants had prior experience with AR applications. We used a within-subject design, with all participants performing the tasks in all conditions. The AR labeling application ran on a 10 inch computer tablet (i.e., Apple iPad). The real-world entity labeled was an anatomical brain model, with twelve AR labels. The task was to locate the label specified at the top left corner of the screen. Once the participant found a label, they would indicate it by touching it on the screen with their index finger. There were eight label location task repetitions, for each of the four conditions. To minimize the influence of confounding factors such as the pose in which each participant holds the tablet with respect to the anatomical brain model, which could result in layout differences, for the pilot study the tablet was placed on a tripod. For the full user study the tablet was handheld by participants. Fig. 5 illustrates the four experimental conditions tested in the pilot study.

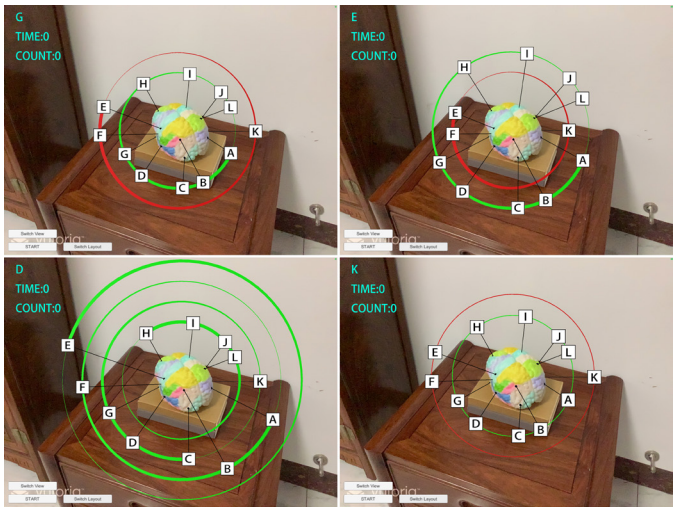

Fig. 5: Screen shots from the four layout visualization configurations tested in the pilot user study: more labels on inner circles (top left), more on outer circles (top right), at most four labels per circle (bottom left), and no line thickness variation (bottom right).

Table 2: Pilot study label finding times, in seconds. Significant differences are indicated with an asterisk.

Condition	Avg $\pm$ std. dev.	Comparison	p
$PEC1$	1.19 ± 0.15		
$PEC2$	1.24 ± 0.13	$PEC2 - PEC1$	0.522
$PEC3$	1.48 ± 0.22	$PEC3 - PEC1$	0.005 *
$PEC4$	1.40 ± 0.18	$PEC4 - PEC1$	0.019 *

Table 2 shows the task completion time for the four conditions. PEC2, PEC3, and PEC4 were compared to PEC1. The normal distribution of the data was confirmed using the Shapiro-Wilk test [43], and the means of PEC2, PEC3, and PEC4 were compared to that of PEC1 using ANOVA. Performance in PEC3 and PEC4 was significantly slower ($p < 0.05$) than in PEC1. Performance was faster in PEC1 compared to PEC2, but the difference was not significant. This confirms our intuition that fewer circles are better (i.e., PEC3 is slower than PEC1), and that the circle line thickness decrease helps (i.e., PEC4 is slower than PEC1). We chose PEC1 over PEC2 since it was somewhat faster.

The conclusion of the pilot study was that our most promising layout visualization is to place as many labels as possible on each circle, and to use a decreasing circle line thickness to indicate the first label and the sorting order on each circle.

4.2 User study

We have conducted a user study to evaluate the most promising configuration of our label layout visualization, with and without presorting, by comparing it to an unsorted single circle layout. We present our study using a format prescribed for health informatics evaluation reports [46].

4.2.1 Study context

We conducted the user study with the approval of our university's Biology and Medical Ethics Committee, which acts as the Institutional Review Board overseeing all human subject studies. The study was conducted during spring 2021, when the acute phase of the COVID-19 epidemic had subsided and human subject studies were possible. The study took the COVID-19 safety measures prescribed by our university, which included wearing face masks, using hand sanitizer, sanitizing the tablet between participants, and social distancing.

Our label layout method was deployed on a 10 inch computer tablet (i.e., Apple iPad), which the participant held with both hands in landscape mode. The AR application was implemented for iOS using the Vuforia AR SDK [40] and Unity [1]. The layout computation algorithm (Algorithm 1) converges in at most 18 iterations. The AR application runs at the tablet maximum display frame rate of 60fps.

4.2.2 Methods

Study design. We have compared our method without presorting (EC1) and with presorting (EC2) to the conventional approach of a single circle layout (CC) in a within subject controlled study. The method was the independent variable of our study defining two experimental conditions (EC1 and EC2), and one control condition (CC). Each participant was exposed to each condition, in random order. For both EC1 and EC2 we used the circle line thickness variation and the greedy assignment of as many labels as possible on each circle, starting from the inner circle, the best configuration from our pilot study.

Hypotheses. Our method was designed to allow a user to locate a label quickly and with little effort leveraging the sorting of the labels on each layout circle. Thus, we formulate the following hypotheses.

H_1 : Users can locate a label *faster* with our partially-sorted concentric layout without presorting (EC1) compared to a conventional single-circle unsorted layout (CC), and the presorting (EC2) allows users to locate labels even faster than when no presorting is used (EC1).

H_2 : User *task load* with EC1 is lower than with CC, and user task load with EC2 is lower than with EC1.

H_3 : EC1 is *easier to use* than CC, and EC2 easier to use than EC1.

Participants. We recruited $N = 34$ participants, of age between 20 and 28, 23 male and 11 female, with good or corrected vision. 11 of our participants had prior experience with AR applications. We expect *large* effect sizes, i.e., with a Cohen's $d > 0.8$ [16, 42], so $N = 34$ provides adequate statistical power.

Study flow. The task was to find eight labels, one at the time, in each of the three conditions, in each of two scenes (Fig. 1): the brain anatomical model, which had a total of 12 labels, and the airplane model, which had a total of 10 labels. The brain and airplane models were 3D printed, so the geometric models of the two scenes are known to the AR application.

The experimental procedure starts with a three minute training period when the participant familiarizes themselves with the task of finding labels and the three types of layouts used in the three conditions. The participant is also told that the green circles have the labels sorted in clockwise order and that the red circles have the labels sorted in counterclockwise order, starting at the thickest part of the circle, and progressing towards the thinnest part. The training is conducted on a computer graphics model of an engine (see Figure 2), such that the participant does not become familiar with the brain and airplane scenes used in the actual experiment.

After the training, each participant had to find 48 labels, one at the time. The label to be found is displayed at the top left corner of the tablet. When the participant finds a label in the layout, they indicate the label location by touching the label text on the tablet screen. After a label is found, the participant also has to touch the label's anchor point. When the anchor point is not visible due to occlusions, the participant has to change the view to establish line of sight to the anchor point. Occlusions were rendered accurately by the AR application using the known geometry of the anatomical brain and of the airplane models. Finding the anchor point requires the user to change the view, which allows testing the label layout in a variety of views.

Outcome measures. We investigated our hypotheses with objective and subjective outcome measures (dependent variables).

We measured the *time* it took to find each label, defined as the time from when the label to be found was displayed on the top left corner of the tablet to the time when the participant touched the label text on the tablet screen. The time to touch the anchor point after finding the label is not included in our time metric. The performance of our label layout visualization method was also measured with two additional dependent variables: the number of layout circles and the number of labels placed on the inner circle. These variables are not directly investigating our hypotheses but they quantify the complexity of the layout.

One subjective outcome measure is task load, for which we used a standard measuring instrument, i.e., the RTLX questionnaire [24], which is a simplified version of the NASA-TLX questionnaire [25]. Finally, we measured usability through a questionnaire. The nine questions were: was the layout easy to use (Q1), was the layout easy to learn (Q2), will you be able to use the layout in the future (Q3), do you think people can easily learn this layout (Q4), was the layout complex (Q5), was the layout confusing (Q6), was the layout reasonable (Q7), was the layout messy (Q8), did the layout help you quickly find the label you were looking for (Q9).

Methods for data acquisition and measurement. The time to find a label was recorded automatically, using a feature built into our AR application. The layout complexity dependent variables were also recorded automatically by our application. The RTLX and the usability questionnaires were filled out by participants in pen and paper form after each condition. All questions were answered with a numerical score from 1 to 10. Some questions are negative, and some are positive, which reduces the risk of mechanical answers.

Methods for data analysis. The time to find labels, the task load, and the usability scores were compared across the three conditions using a one-way repeated-measures ANOVA. First, the distribution normality assumption was verified using the Shapiro-Wilk test [43]. All our data satisfied the normality assumption. Then the sphericity assumption is evaluated using the Mauchly test [36]. When the sphericity assumption is violated, a Greenhouse-Geisser correction is applied to the data. Then an overall ANOVA was conducted to investigate whether one can reject the null hypothesis that there is no statistically significant difference between the three conditions. When the null hypothesis was rejected ($p < 0.05$), the differences between the three pairs was analyzed with post-hoc tests, with a significance level lowered conservatively ($\alpha < 0.016$) using the Bonferroni correction. For the time dependent variable we also quantified effect magnitude using Cohen's d [16], with the customary qualitative labeling of the numerical values, i.e., "huge" for $d > 2.0$, and "very large" for $2.0 > d > 1.2$. The label finding times were aggregated per participant, condition, and scene before analysis, i.e., each participant data was reduced to six averages, one per condition and scene pair, before being input into the ANOVA analysis. The statistical analysis was performed using the SPSS software [28].

4.2.3 Results

All $N = 34$ participants completed the experiment without unexpected events, so no participant data was discarded.

Table 3: Time to find labels, in seconds: descriptive statistics and pairwise comparisons between conditions. Statistical significance ($p < 0.016$) is denoted with an asterisk.

Scene	Cond.	Avg ± std. dev.	Pairwise comparisons	p	Cohen's d	Effect size
Brain	CC	4.7 ± 0.3	$CC - EC1$	$< 0.001*$	2.5	Huge
	$EC1$	3.5 ± 0.6	$CC - EC2$	$< 0.001*$	5.9	Huge
	$EC2$	2.3 ± 0.5	$EC1 - EC2$	$0.001*$	2.1	Huge
Airplane	CC	3.2 ± 0.4	$CC - EC1$	0.060	1.4	V. large
	$EC1$	2.8 ± 0.2	$CC - EC2$	$< 0.001*$	3.6	Huge
	$EC2$	2.1 ± 0.2	$EC1 - EC2$	$< 0.001*$	3.7	Huge

The label finding times and the pairwise comparisons between conditions are given in Table 3. The box plots are given in Fig. 6 (left). The sphericity assumption is verified: $p = 0.787$ (brain) and $p = 0.054$ (airplane). The overall ANOVA reveals that there is a statistically significant difference between the three conditions for the brain scene ($F(2, 66) = 74.46$, $P < 0.001$) and for the airplane scene ($F(2, 66) = 27.75$, $P < 0.001$). Post-hoc analysis reveals that EC1 label finding times were significantly shorter than for CC, that EC2 times were significantly shorter than for EC1, and that EC2 times were significantly shorter than for CC, for both scenes, except for CC versus EC1 for the airplane scene. The effect size was "Huge" for all pairwise comparisons except for the CC-EC1 for the airplane, where it was "Very large".

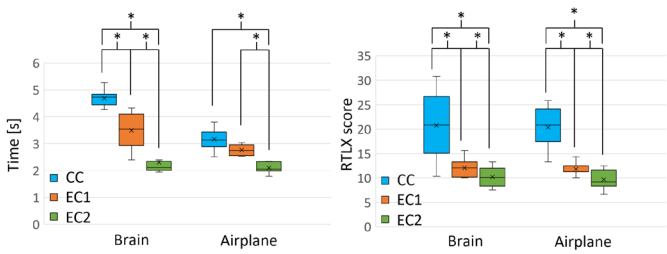

Fig. 6: Box plots for label finding times (left) and task load scores (right), for the three conditions and the two scenes. Asterisks denote statistical significance.

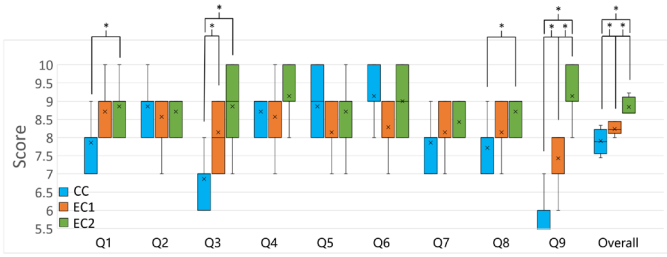

Fig. 7: Usability questionnaire results for individual questions and over all questions. Significant differences are denoted with an asterisk. Scores of questions about negative features are complemented such that a higher score always means better.

Table 4: Layout statistics.

Scene	Condition	Avg. circles	Min. circles	Max. circles	Labels on inner circle
Brain	$EC1$	2.97	2	3	5.7
	$EC2$	1.99	1	2	9.8
Airplane	$EC1$	3.00	3	3	6.0
	$EC2$	2.00	2	2	8.7

Table 4 gives the number of circles in our EC1 and EC2 layouts, for the two scenes. Presorting the labels almost always reduces the number of circles from 3 to 2, which explains the significant advantage of EC2 over EC1. There are never more than 3 circles (Max. circles=3), and EC2 occasionally places all labels on a single circle (Min. circles=1). Table 4 also gives the average number of labels on the inner circle, which shows that EC2 places most of the 12 labels of the brain and most of the 10 labels of the airplane on the inner circle of the layout.

The RTLX task load scores are shown in Fig. 6 (right). The sphericity assumption is violated: $p = 0.001$ (brain) and $p < 0.001$ (airplane). After applying the Greenhouse-Geisser correction, the overall ANOVA reveals significant differences between the three conditions: ($F(1.16, 37.11) = 67.38$, $P < 0.001$) for the brain scene, and ($F(1.11, 39.98) = 108.90$, $P < 0.001$) for the airplane scene. The post-hoc analysis reveals that all pairwise differences are significant in favor of CC over EC1 and EC2, and of EC1 over EC2, for both scenes ($p < 0.001$ for all six comparisons).

The answers to the individual questions as well as the average over all questions are given in Figure 7. A negative score x is replaced with its complement 11-x such that larger is always more favorable. Over all nine questions, the sphericity assumption is validated ($p = 0.322$), the overall ANOVA reveals significant differences between the three conditions ($F(2, 66) = 26.83, p < 0.001$), and post-hoc analysis reveals that EC2 is significantly easier to use than CC and than EC1, and

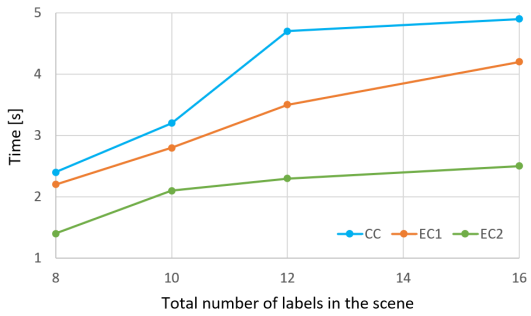

Fig. 8: Label finding time dependency on the number of labels.

that EC1 is significantly easier to use than CC. In terms of individual questions, statistically significant pairwise differences were recorded for Q1, Q3, Q8, and Q9. The scores for the four questions satisfy the sphericity assumption ($p = 0.117$, $p = 0.558$, $p = 0.994$, and $p = 0.093$), and the overall ANOVA reports significant differences between the three conditions ($F(2,66) = 4.691$, $p = 0.031$; $F(2,66) = 12.411$, $p = 0.001$; $F(2,66) = 5.550$, $p = 0.020$; $F(2,66) = 26.78$, $p < 0.001$).

4.2.4 Discussion

The results support hypothesis H_1. Our label layout leads to faster label finding compared to a conventional single-circle unsorted layout. Furthermore, presorting helps reduce the number of layout circles and therefore it accelerates label finding even more. Five of the six differences are statistically significant (Table 3, Fig. 6). The only exception is for EC1 vs. CC for the airplane scene, where the total number of labels is small enough to not make CC exceedingly slow, yet the geometric complexity is high enough for the unsorted labels to always in three circles for EC1 (Table 4). Presorting reduces the number of circles to two, which speeds up label finding significantly.

We have extended the user time study data (Table 3, Fig. 6 (left)) in an additional experiment aimed to elucidate the dependency of time to the total number of labels. We asked a subset of 10 of our original participants to find labels in the brain scene where the total number of labels was changed to 8, 10, and 16, providing three additional data points (in the main user study the brain scene had 12 labels). Fig. 8 shows that the label finding time grows more slowly with the number of labels in the scene for EC2 and EC1 compared to CC, for up to 12 labels. EC1 and EC2 maintain their advantage over CC for 16 labels, but the advantage is smaller. This is likely to be a consequence (1) of the larger number of circles needed for our layouts, which reduces the efficiency of EC1 and EC2, and (2) of the ability to consider several labels in parallel as the density of labels increases in CC. We conclude that our approach is suitable for scenes with up to 16-20 labels which cap the maximum number of circles to 3 (Table 1), and provide faster label finding than CC (Fig. 8). As the number of labels increases, the CC approach also becomes intractable. Larger sets of labels have to be first pruned to a relevant subset using orthogonal approaches such as color coding, eye tracking, and intelligent filtering.

The results (Fig. 6) (right) support hypothesis H_2, with significantly lower task loads for EC2 vs EC1 and for EC1 vs CC, for both scenes. As expected, users can find labels more easily on the sorted circles compared to the unsorted circle, and fewer circles are better.

The results support hypothesis H_3 when the answers to all nine questions of the usability questionnaire are aggregated (Fig. 7). In terms of individual questions, all significant differences were in favor of our layouts when compared to the conventional layout. Not all differences were significant, and a larger study might be needed to increase statistical power and establish significance. The only two questions where CC had an advantage (that was not statistically significant) were asking participants whether they found the layout complex (Q5) and confusing (Q6). This is expected since, indeed, reading our layouts requires familiarity with the increased structure of the layouts, which have multiple circles, with the labels of a circle being sorted, and with the color of the circle line and the change in line thickness indicating the sorting order. The CC layout is unstructured and the participant can straightforwardly examine the circle sequentially. As established earlier, the additional structure in our layouts leads to faster label finding. Users are able to learn the structure quickly and to use it successfully.

5 Conclusions, Limitations, Future Work

We have presented a method for displaying labels in AR applications that makes it easier for the user to find a specific label. The essence of our method is to leverage a sorting of the labels, while avoiding label and leader line intersections. Our layout uses a small number of concentric circles, with the labels on each circle being sorted. The complexity of our layout is bound by a result from circular permutation research that prescribes the minimum length of the longest sorted subsequence of labels. An optional presorting step further reduces the average number of circles. In a user study, our layout compared favorably to the conventional approach of a single circle layout, reducing the average time participants needed to locate a label.

One limitation of our method is that the multiple circle layout requires more screen real estate than a single layout, which means that less space is available to visualize the real world object of interest to the application. In the AR context, this means that the user has to hold the tablet sufficiently far away from the object to leave room around the object for the labels. We note that in the figures shown in this paper, e.g., Figure 1, the labels are drawn larger than they are on the tablet, to make the figure readable in the paper. We refer the reader to the accompanying video and to the supplemental material which show screen captures where the labels are drawn with their accurate relative size. A related limitation is that, like all label visualization methods, our approach prefers short label text strings, to avoid obscuring too much of the object. Compared to the single circle approach, our method is more robust to label text length as the crowding on each circle is reduced due to the smaller number of labels per circle."

Our method applies to the outside-looking-in visualization scenario, with a single object of interest, that can be surrounded by its labels. Future work should examine the more challenging inside-looking-out visualization scenario, with the user immersed in the scene of interest, which might require partially sorting labels on more general closed curves, not just concentric circles. As stated in the prior work Sect. 2.4, in addition to leader lines, there are other approaches for catering to and guiding the user's attention, and future work could investigate multi-step approaches where color coding, eye tracking, or intelligent filtering are used first to identify the subset of labels of interest, which are then laid out using our approach.

Although our method was presented and evaluated in the context of AR, our label layouts might prove to be useful in the context of other interactive visualization contexts. From an implementation standpoint, our method readily works for fully virtual scenes. Our method is particularly useful in contexts such as AR and VR that rely on HMD visualization and where the user cannot easily rely on a keyboard to search for the label textually. One direction of future work is to deploy our method to an HMD AR interface. An ideal AR HMD that does not reduce the user's natural field of view would avoid the screen real estate limitation mentioned above for our tablet AR interface. However, an AR HMD that limits the user's field of view will require that the user trace the circle with the central part of their field of view to be able to read the labels. This problem is not unique to our method and the circle could actually help guide the user towards the labels that are clipped by the small active field of view. Future studies should also quantify the benefit of our method in VR, which has the advantage of HMDs with a wider active field of view, providing more visualization real estate.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China through Projects 61932003 and 61772051, by National Key RD plan 2019YFC1521102, by the Beijing Natural Science Foundation L182016, by the Beijing Program for International ST Cooperation Project Z191100001619003, by the funding of Shenzhen Research Institute of Big Data(Shenzhen 518000).

REFERENCES

[1] Unity. https://unity3d.com.

[2] K. Ali, K. Hartmann, and T. Strothotte. Label layout for interactive 3d illustrations. *Journal of WSCG*, 13(1):1–8, 2005.

[3] R. Azuma and C. Furmanski. Evaluating label placement for augmented reality view management. In *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pp. 66–75. IEEE, 2003.

[4] J. Balata, L. Cmolik, and Z. Mikovec. On the selection of 2d objects using external labeling. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2255–2258, 2014.

[5] L. Barth, A. Gemsa, B. Niedermann, and M. Nöllenburg. On the readability of leaders in boundary labeling. *Information Visualization*, 18(1):110–132, 2019.

[6] M. A. Bekos, S. Cornelsen, M. Fink, S.-H. Hong, M. Kaufmann, M. Nöllenburg, I. Rutter, and A. Symvonis. Many-to-one boundary labeling with backbones. *J. Graph Algorithms Appl.*, 19(3):779–816, 2015.

[7] M. A. Bekos, M. Kaufmann, M. Nöllenburg, and A. Symvonis. Boundary labeling with octilinear leaders. *Algorithmica*, 57(3):436–461, 2010.

[8] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Polygon labelling of minimum leader length. In *Asia-Pacific Symposium on Information Visualisation*, pp. 15–21, 2006.

[9] M. A. Bekos, B. Niedermann, and M. Nöllenburg. External labeling techniques: A taxonomy and survey. In *Computer Graphics Forum*, vol. 38, pp. 833–860. Wiley Online Library, 2019.

[10] B. Bell, S. Feiner, and T. Höllerer. View management for virtual and augmented reality. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pp. 101–110, 2001.

[11] S. Bianco, F. Gasparini, and R. Schettini. Color coding for data visualization. In *Encyclopedia of Information Science and Technology, Third Edition*, pp. 1682–1691. IGI Global, 2015.

[12] E. Castellina and F. Corno. Multimodal gaze interaction in 3d virtual environments. *Cogain*, 8(2008):33–37, 2008.

[13] L. Čmolík and J. Bittner. Real-time external labeling of ghosted views. *IEEE transactions on visualization and computer graphics*, 25(7):2458–2470, 2018.

[14] L. Cmolik, V. Pavlovec, H.-Y. Wu, and M. Nollenburg. Mixed labeling: Integrating internal and external labels. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[15] E. M. Coelho, S. Julier, and B. Maclntyre. Osgar: A scene graph with uncertain transformations. In *Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 6–15. IEEE, 2004.

[16] J. Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

[17] É. Czabarka and Z. Wang. Erdős–szekeres theorem for cyclic permutations. *Involve, a Journal of Mathematics*, 12(2):351–360, 2018.

[18] P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compositio mathematica*, 2:463–470, 1935.

[19] J.-D. Fekete and C. Plaisant. Excentric labeling: Dynamic neighborhood labeling for data visualization. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 512–519, 1999.

[20] A. Gemsa, J.-H. Haunert, and M. Nöllenburg. Multirow boundary-labeling algorithms for panorama images. *ACM Transactions on Spatial Algorithms and Systems*, 1(1):1–30, 2015.

[21] T. Götzelmann, K. Hartmann, and T. Strothotte. Agent-based annotation of interactive 3d visualizations. In *International Symposium on Smart Graphics*, pp. 24–35. Springer, 2006.

[22] R. Grasset, T. Langlotz, D. Kalkofen, M. Tatzgern, and D. Schmalstieg. Image-driven view management for augmented reality browsers. In *2012 IEEE International Symposium on Mixed and Augmented Reality*, pp. 177–186. IEEE, 2012.

[23] Q. Han, M. John, S. Koch, I. Assenov, and T. Ertl. Labeltransfer-integrating static and dynamic label representation for focus+ context text exploration. In *2018 International Symposium on Big Data Visual and Immersive Analytics*, pp. 1–8. IEEE, 2018.

[24] S. G. Hart. Nasa-task load index (nasa-tlx); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, pp. 904–908. Sage Publications Sage CA: Los Angeles, CA, 2006.

[25] S. G. Hart and L. E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, vol. 52, pp. 139–183. Elsevier, 1988.

[26] K. Hartmann, K. Ali, and T. Strothotte. Floating labels: Applying dynamic potential fields for label layout. In *International Symposium on Smart Graphics*, pp. 101–113. Springer, 2004.

[27] K. Hartmann, T. Götzelmann, K. Ali, and T. Strothotte. Metrics for functional and aesthetic label layouts. In *International Symposium on Smart Graphics*, pp. 115–126. Springer, 2005.

[28] IBM. Spss software. https://www.ibm.com/analytics/spss-statistics-software.

[29] P. Kindermann, B. Niedermann, I. Rutter, M. Schaefer, A. Schulz, and A. Wolff. Multi-sided boundary labeling. *Algorithmica*, 76(1):225–258, 2016.

[30] D. Kouřil, L. Čmolík, B. Kozlíková, H.-Y. Wu, G. Johnson, D. S. Goodsell, A. Olson, M. E. Gröller, and I. Viola. Labels on levels: labeling of multi-scale multi-instance and crowded 3d biological environments. *IEEE transactions on visualization and computer graphics*, 25(1):977–986, 2018.

[31] M. Lankes and B. Stiglbauer. Gazear: Mobile gaze-based interaction in the context of augmented reality games. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pp. 397–406. Springer, 2016.

[32] G. Li, Y. Liu, and Y. Wang. An empirical evaluation of labelling method in augmented reality. In *Proceedings of the 16th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and its Applications in Industry*, pp. 1–9, 2018.

[33] M. Löffler, M. Nöllenburg, and F. Staals. Mixed map labeling. In *International Conference on Algorithms and Complexity*, pp. 339–351. Springer, 2015.

[34] J. B. Madsen, M. Tatzqern, C. B. Madsen, D. Schmalstieg, and D. Kalkofen. Temporal coherence strategies for augmented reality labeling. *IEEE transactions on visualization and computer graphics*, 22(4):1415–1423, 2016.

[35] J. Marks and S. M. Shieber. The computational complexity of cartographic label placement. *Harvard Computer Science Group Technical Report TR-05-91.*, 1991.

[36] J. W. Mauchly. Significance test for sphericity of a normal n-variate distribution. *The Annals of Mathematical Statistics*, 11(2):204–209, 1940.

[37] S. Oeltze-Jafra and B. Preim. Survey of labeling techniques in medical visualizations. In *Proceedings of the 4th Eurographics Workshop on Visual Computing for Biology and Medicine*, pp. 199–208, 2014.

[38] J. Orlosky, K. Kiyokawa, T. Toyama, and D. Sonntag. Halo content: Context-aware viewspace management for non-invasive augmented reality. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pp. 369–373, 2015.

[39] H. M. Park, S. H. Lee, and J. S. Choi. Wearable augmented reality system using gaze interaction. In *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 175–176. IEEE, 2008.

[40] PTC. Vuforia developer portal. https://developer.vuforia.com.

[41] T. Ropinski, J.-S. Praßni, J. Roters, and K. H. Hinrichs. Internal labels as shape cues for medical illustration. In *Vision Modeling and Visualization*, vol. 7, pp. 203–212. Citeseer, 2007.

[42] S. S. Sawilowsky. New effect size rules of thumb. *Journal of Modern Applied Statistical Methods*, 8(2):26, 2009.

[43] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965.

[44] F. Shibata, H. Nakamoto, R. Sasaki, A. Kimura, and H. Tamura. A view management method for mobile mixed reality systems. In *Immersive Projection Technology Workshop/Eurographics Symposium on Virtual Environments*, pp. 17–24, 2008.

[45] T. Stein and X. Décoret. Dynamic label placement for improved interactive exploration. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*, pp. 15–21, 2008.

[46] J. Talmon, E. Ammenwerth, J. Brender, N. De Keizer, P. Nykänen, and M. Rigby. Stare-hi—statement on reporting of evaluation studies in health informatics. *International journal of medical informatics*, 78(1):1–9, 2009.

[47] M. Tatzgern, D. Kalkofen, R. Grasset, and D. Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3d space. In *2014 IEEE Virtual Reality*, pp. 27–32. IEEE, 2014.

[48] M. Tatzgern, D. Kalkofen, and D. Schmalstieg. Dynamic compact visualizations for augmented reality. In *2013 IEEE Virtual Reality*, pp. 3–6. IEEE, 2013.

[49] M. Tatzgern, V. Orso, D. Kalkofen, G. Jacucci, L. Gamberini, and D. Schmalstieg. Adaptive information density for augmented reality displays. In *2016 IEEE Virtual Reality*, pp. 83–92. IEEE, 2016.