Mixed Labeling: Integrating Internal and External Labels

Ladislav Čmolík, Václav Pavlovec, Hsiang-Yun Wu, and Martin Nöllenburg

Abstract—In this paper, we present an algorithm capable of mixed labeling of 2D and 3D objects. In mixed labeling, the given objects are labeled with both internal labels placed (at least partially) over the objects and external labels placed in the space around the objects and connected with the labeled objects with straight-line leaders. The proposed algorithm determines the position and type of each label based on the user-specified ambiguity threshold and eliminates overlaps between the labels, as well as between the internal labels and the straight-line leaders of external labels. The algorithm is a screen-space technique; it operates in an image where the 2D objects or projected 3D objects are encoded. In other words, we can use the algorithm whenever we can render the objects to an image, which makes the algorithm fit for use in many domains. The algorithm operates in real-time, giving the results immediately. Finally, we present results from an expert evaluation, in which a professional illustrator has evaluated the label layouts produced with the proposed algorithm.

Index Terms—Labeling, Mixed labeling, Internal labeling, External labeling, Expert evaluation.

1 Introduction

Graphics such as illustrations, data visualizations, and information graphics are designed to communicate information visually. However, in most cases, the graphics cannot convey the whole information themselves. Therefore, the visual information is typically accompanied by verbal information in the form of text or audio. In such cases, labels, short textual annotations, that mediate the connection between the visual and verbal information, play an essential part in the design of a graphic.

The label layout, i.e., the positioning of the labels, plays a crucial role in the efficient and correct understanding of the communicated information. According to Tufte [1], label layouts should not use legends but embed all the necessary text into the graphics itself.

A convenient and functional label layout has to exhibit four general characteristics: Readability, unambiguity, compactness, and aesthetics [2]. More specifically, all labels should be readable without occlusions. The viewer should be able to easily associate the labels to the labeled objects and vice versa. The label layout should use as little space around the illustration as possible. This characteristic is essential, especially when we embed graphics on a page of text. Finally, the label layout should be pleasing to the readers' eyes. However, we should keep in mind that the aesthetics are most often subjective.

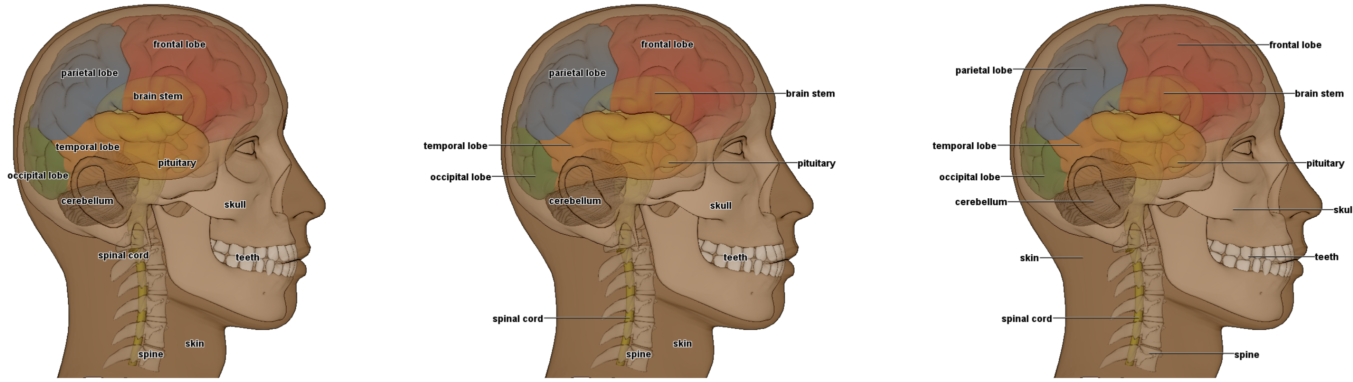
In this paper, we are focusing on the labeling of area features, where we can divide labels into two categories based on their positioning: *internal labels* are overlapping the labeled objects, at least partially, while *external labels* are typically not overlapping the labeled objects and are connected with the labeled objects by leaders. A leader can be a straight-line, a polyline, or a smooth curve. Figure 1 shows label layouts using internal labels and/or external labels with straight-line leaders.

Various types of labels are utilized in various domains. Technical illustrations and encyclopedia illustrations almost exclusively use external labels [3]. On the other hand, illustrations in medical atlases [4] use both internal and external labels, where the internal labels are entirely inside of the labeled areas. In cartography and data visualizations, area features are labeled with both internal and external labels [5], but internal labels are allowed to overlap the labeled areas only partially if they maintain an unambiguous association with the labeled areas (e.g., small islands in maps or glyphs in data visualizations). Generally, internal positions are preferred in maps and information graphics, but if the features are locally densely packed and there is a lack of space, illustrators switch to external labels.

Most of the previous work, discussed in detail in Section 2, is focusing solely on internal or external labels. Only a few methods are using both internal and external labels in a single label layout. However, these methods determine positions of internal labels independently from external labels and vice versa. Such approaches may lead to overlaps of leaders with internal labels. Further, they require that every internal label is positioned entirely inside of its area, which excludes label layouts, where the internal labels overlap the labeled objects only partially; such label layouts, however, are useful in data visualization and microbiology [6].

In this work, we propose a more flexible approach to the mixed labeling of area features that is able to use both internal and external labels in one label layout. We highlight our three main contributions:

- *Ladislav Čmolík is with Faculty of Electrical Engineering at CTU in Prague, Prague, Czechia. E-mail: cmolikl@fel.cvut.cz.*
- *Václav Pavlovec is with Faculty of Electrical Engineering at CTU in Prague, Prague, Czechia. E-mail: pavlova1@fel.cvut.cz.*
- *Hsiang-Yun Wu is with Institute of Visual Computing and Human-Centered Technology at TU Wien, Vienna, Austria. E-mail: hsiang.yun.wu@acm.org.*
- *Martin Nöllenburg is with Institute of Logic and Computation at TU Wien, Vienna, Austria. E-mail: noellenburg@ac.tuwien.ac.at.*



(a) (b) (c)

Fig. 1. (a) 3D model of a human head with an internal label layout created with the proposed method. (b) By changing the value of the ambiguity threshold t_a, the user can create a mixed label layout where external labels are used instead of the internal labels with possibly ambiguous placement (e.g., the spinal cord label). (c) By setting the value of the threshold t_a to the maximum value, all labels are positioned externally.

1) We propose an internal labeling algorithm to compute label layouts. Internal labels are allowed to overlap the labeled objects fully or only partially while maintaining an unambiguous association with the labeled objects whenever possible. The objects can have any shape, including non-convex shapes. To achieve this, we present new criteria designed to prioritize positions with an unambiguous association between labeled objects and internal labels. Our algorithm is able to label also overlapping areas. We label 3D models with semitransparent objects to demonstrate this ability.
2) To achieve mixed labeling with both internal and external labels, we show how to integrate the modified external labeling algorithm of Čmolík and Bittner [7] into the proposed internal labeling algorithm. We have modified their external labeling algorithm to allow external labeling of objects of non-convex shapes and to prioritize positions with an unambiguous association between labeled objects and external labels.
The mixed labeling algorithm determines label layouts, where the labels do not overlap, and the straight-line leaders of external labels do not cross internal labels. The user can control the algorithm by setting the ambiguity threshold t_a to force the method to use external labels instead of internal labels if they would have an ambiguous association with the labeled objects. See Figure 1.
3) The proposed mixed labeling algorithm is a screen-space technique; it functions in an image with encoded 2D objects, as well as projections of 3D objects. Consequently, we can use the algorithm whenever we can render objects into an image, making it suitable for application in many domains. The algorithm functions in real-time, providing the results instantly. The real-time performance allows users to interact with the scene (e.g., pan, zoom, rotate). However, the algorithm does not produce temporally coherent label layouts [8]. Therefore, we do not show the label layout during user interaction.

2 Related Work

We divide the related work according to the positioning of the labels into internal, external, and mixed labeling methods. A lot of the labeling literature considers labeling of point features, but here we only mention those that are sampling a representative point per area feature to label area features. Primarily, our focus is on methods specifically designed for labeling of area features.

2.1 Internal Labeling Methods

In many domains, internal labels are the preferred style of labeling area features. Cartography is a domain with vast experience and established guidelines for internal label placement of area features. Yoeli [5] recommends that a label should be placed internally if its not occluding central parts of other areas. Further, the internal label should overlap the most central part of the labeled area and fit inside the area if possible. Note that a label that fits inside the labeled area may still occlude central parts of other areas if the areas are not mutually exclusive (e.g., when we label semitransparent objects). Further, to fit an internal label to the labeled 2D area, the label text is allowed to follow the shape of the labeled area [9].

A few automated approaches following the general cartographic placement guidelines have been developed in the cartography domain. Van Roessel [10] presents an algorithm for computing label candidates for axis-aligned rectangles in a given polygonal area as needed for area labeling in maps. Barrault [11] describes a fitness measure for candidate positions of shape-fitted area labels and a corresponding label selection method. Freeman [12] sketches a general approach and guidelines for labeling point, line, and area features, but no specific algorithms are given.

When internal labels are used to annotate surfaces of 3D objects, the labels often follow the shape of the 3D surfaces. Ropinski et al. [13] are using 3D shape fitting to annotate surfaces of 3D models for medical illustrations. Cipriano and Gleicher [14] introduce a special text scaffold surface that is computed on top of the given 3D model to avoid occlusion and distortion of the labels of medical and microbiological 3D models. Prado et al. [15] are projecting multiple copies of labels directly onto the objects in the 3D scene. Maass and Döllner [16] integrate labels onto important objects (e.g., buildings) in 3D virtual landscapes.

In all of the approaches mentioned above, the labels are required to fit into their mutually disjoint areas. One approach where the internal labels are overlapping the labeled areas only partially is the approach of Kouřil et al. [6], where they place labels for hierarchically organized area features in interactive 3D models. They determine a representative anchor point for each area and use billboard labels with the anchor point at its center. However, they do not provide a mechanism to prevent overlaps of labels.

2.2 External Labeling Methods

In external labeling, labels are usually connected to their features via additional leaders, which can be straight-line, polyline, or smooth curves. This is the predominant style in highly detailed technical and medical illustrations, where text should not occlude important features of the background image [17].

To apply external labeling for area features, one can either determine a representative point inside each feature and then use a point-labeling method (see the recent survey of Bekos et al. [17] for an overview) or use an algorithm that combines the selection of a suitable leader endpoint together with the leader and label placement. Some methods also place external labels in the direct vicinity of area features, e.g., islands in a map, by first generating and evaluating candidate positions and then using simulated annealing for label optimization [18].

Many algorithms for external labeling actually consider a bounding box of the illustration and place the labels on its boundary; this is known as *boundary labeling*. Exact algorithms, typically minimizing the total leader length for a given set of point features and unit-height labels, are known for different leader shapes and placement of labels on the different sides of the bounding box [17]. Most of the algorithms use dynamic programming. The more bounding box sides are used for the labels simultaneously, the more the solution space grows, and thus the more complex the algorithms get. While many algorithms use pre-defined but exchangeable label positions, others allow moving the labels along the boundary to find the best positions [19], [20]. Preim et al. [21] consider straight-line leaders and temporally consistent labels for interactive illustrations, although this can result in intersecting leaders. Some boundary labeling algorithms are specifically designed for area features. Bekos et al. [22] minimize the length of crossing-free polyline leaders over all possible anchor points within the given set of area features using an exact, matching-based algorithm. Bekos et al. [23], as well as Löffler et al. [24], use two types of labels for point features: labels that are close to the points and do not need a leader and external labels with a leader. They present exact algorithms, where the objective is to maximize the number of internally labeled points, while the remaining points are labeled externally on one side of the illustration using leaders. Please note that these methods are designed for point features, and there is no imediate generalization to area features, as the established guidelines for area feature and point feature labeling differ.

For more general image contours, e.g., a convex hull, an enclosing circle or some other convex shape that is enclosing all labeled objects, most algorithms apply straight-line leaders. Ali et al. [2] describe a variety of external labeling algorithms in this general setting using local optimization techniques. Čmolík and Bittner [7], [25] propose a real-time greedy method for labeling interactive 3D models along a convex contour with different leader types. Niedermann et al. [26] place labels with radially monotone cost-minimal straight-line leaders around convex contours using dynamic programming. Techniques for *excentric* labeling define a (circular) focus lens and arrange labels of features inside the lens along the lens boundary [27].

For even more general image contours, e.g., silhouettes of the labeled objects, Stein and Décoret [28] place label boxes with straight-line leaders in the free space of complex scenes; Wu et al. [29] present an approach to place text labels and images for annotating metro maps without intersecting the individual metro lines. They use external labels without leaders where possible and external labels with straight-line leaders in the free space where necessary. Maass and Döllner [30] use billboard labels with vertical leaders to connect anchors to distant labels in virtual landscapes, but not strictly placing the labels outside the image, whereas Gemsa et al. [31] optimize the placement of the same type of labels above the image.

In our approach, we use a part of the approach of Čmolík and Bittner [7], [25].

2.3 Mixed Labeling Methods

Neither exclusively internal nor exclusively external label layouts for area features provide a satisfying solution for many real-world labeling problems. While the former fail in situations dealing with objects that are smaller than their labels, the latter often waste space and introduce labels that are unnecessarily far away from their features due to not permitting any internal labels. Therefore, in the most general case, label layouts can be composed of a mix of internal and external labels mitigating the aforementioned issues. Bell et al. [32] present a view management system for VR and AR applications, in which area objects are labeled internally, if there is sufficient space, or otherwise, possibly, an external label is placed in the free space using a front-to-back greedy placement. Götzelmann et al. [9], [33], [34] also present real-time methods for labeling interactive 3D illustrations with both internal and external labels. Luboschik et al. [35] present a fast heuristic for labeling point, line, and area features that selects greedily the locally best available position for each label, starting with internal labels and proceeding to external labels if necessary. For the sake of speed, some aesthetic trade-offs are made, e.g., leaders may cross.

The above methods divide the labeled objects into two groups, where one group is labeled internally, and the second group is labeled externally. The label layout for each group is determined independently from the other group. Such an approach leads to potential overlaps of the leaders of external labels with the internal labels. The strict separation into internal and external labels also discards all labels that are only partially inside an object, but could still be associated easily with the object. As a consequence, it is impossible to label small objects with long labels internally. When these labels are all positioned fully externally, the resulting label layout may become unnecessarily large.

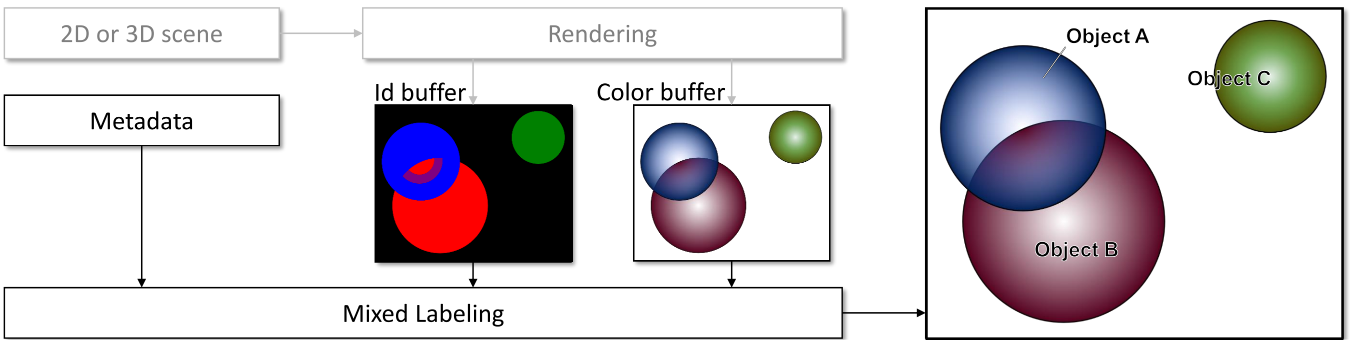


Fig. 2. Overview of the proposed method. The method takes an id buffer, color buffer synthesized from the scene, and metadata in the form of short annotations as the input. The method determines the label layout based on the information encoded in the id buffer and overlays the color buffer with the label layout. Please see the supplementary material for graphical overview of the first two steps of the algorithm with all used buffers.

Positioning the labels partly inside and partly outside of the objects gives us more flexibility in the label layout and typically also yields a more compact layout.

3 Our Approach to Mixed Labeling

In this section, we present our approach to the mixed labeling of area features. Unlike the state-of-the-art methods [9], [33], [34], our approach can position internal labels partly outside of the areas of the labeled objects and eliminate the overlaps of the labels. The user is able to control the allowed ambiguity of the internal labels with the ambiguity threshold t_a. The internal labels that would be placed on positions with ambiguity greater than the given threshold are placed externally instead. For external labels, we use straight-line leaders (also denoted as *leader lines*), which have been shown to be one of the two most readable leader types (together with 1-bend orthogonal polylines) by Barth et al. [36]. Our approach further eliminates overlaps of internal labels with external labels or leader lines. The positions of the external labels are again determined to minimize the ambiguity of the association between labels and labeled objects.

3.1 Overview of the Proposed Mixed Labeling Method

The proposed method is a screen-space technique operating in an image space where the 2D objects or the projected 3D objects are encoded. In other words, the technique is working with *buffers*, i.e., 2D raster images allowed to store other information than just the color for each pixel.

Our method takes two buffers that encode the properties of the objects to be labeled as an input, see Figure 2. The *color buffer* contains the color of the objects, and the *id buffer* contains unique ids of the objects. A further input of the method is metadata in the form of short textual annotations. Our method requires the annotation for each unique id in the *id buffer* as the input.

We denote each region in the *id buffer* with a unique id as an *area* of one of the objects. The number n of unique ids in the *id buffer* gives us the set $\mathbb{A} = \{A_1, \dots, A_n\}$ containing all n areas to be labeled.

To support the labeling of semi-transparent objects, where the areas of the objects are not mutually disjoint and may overlap, we represent the id of one area in the *id buffer* as one bit in the pixel of the buffer. We use an unsigned integer RGBA buffer with 32 bits per channel for the *id buffer*, which allows us to store 128 ids in one pixel. In other words, the *id buffer* can contain up to 128 overlapping areas of the objects, which was sufficient for our experiments. If one needs to store more areas in the *id buffer*, then one can use multiple RGBA buffers to represent the *id buffer*.

We expect that the rendering method providing the *color buffer* and *id buffer* is using the approach of Čmolík and Bittner [7] to discard ids in regions of the areas where the objects are too transparent, or other almost opaque objects occlude them. This is the case in Figure 2, where some parts of the intersection of Object A (blue) and Object B (red) are assigned exclusively to one object, whereas only the violet part of the intersection is assigned to both objects.

Determining the label layout for a configuration of objects encoded in the *id buffer* is an optimization task. In our proposed method, we use heuristics and a greedy algorithm to determine the label layout. Here, we describe the overview of our method first and explain the details in the following sections, as referenced in parentheses below:

1) Establish internal label candidates and external label candidates for each area $A_i \in \mathbb{A}$. (3.2)
2) Establish buffers for the labeling criteria. (3.4)
3) While there is an unlabeled area in $\mathbb{A}$:
    a) Select the unlabeled area with the lowest *capacity*, indicating the quality of label candidates, as the area A_S for labeling. (3.5)
    b) Find the internal label candidate with maximum *fitness* as the internal label for the selected area A_S. (3.6)
    c) If the *fitness* of the best internal label candidate is lower than the ambiguity threshold t_a
        i) Find the external label candidate with maximum *fitness* as the external label for the selected area A_S.
        ii) If the best external candidate exists then discard internal and external label candidates of yet unlabeled areas that intersect with the determined external label.
    d) Otherwise discard all internal and external label candidates of yet unlabeled areas that intersect with the determined internal label. (3.7)
4) Render the labels over the *color buffer*.

3.2 Establishing Label Candidates

To establish both internal and external label candidates, we first determine the dimensions $\mathbf{d}_i = (w_i, h_i)$ of the label for each area A_i from the provided textual annotations, where w_i is the width, and h_i is the height of the label. We create a list of the dimensions $\mathbb{D} = \{\mathbf{d}_1, \ldots \mathbf{d}_n\}$. We also determine the maximal width w_{max} and the maximal height h_{max} of all the label dimensions.

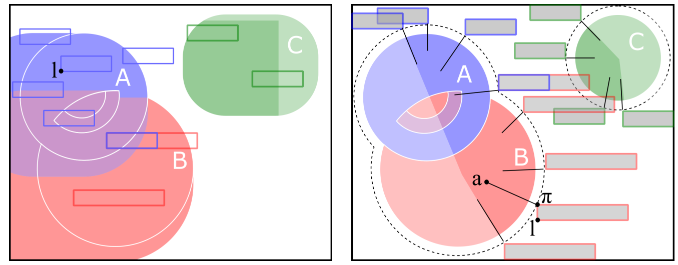
A label candidate is representing one possible position of a label placed over the *color buffer*. We represent one label candidate as one pixel of a buffer with the same resolution as the *color buffer* and *id buffer*. This way, we can evaluate the *fitness* of all label candidates in parallel and store the results in a 2D buffer of positions that correspond to the positions of the label candidates.

We represent each internal label candidate $\mathbf{c_I}$ as the pixel on the position of the lower-left corner $\mathbf{l}$ of the *label box*, which encloses the label. Therefore, we establish the internal label candidates of each area A_i in the *id buffer* by dilating A_i to the left by the width w_i and downwards by the height h_i of the label and storing them in the *internal candidates buffer*. This way, the label box of each internal label candidate of an area A_i will overlap at least one pixel of the area. Note that one pixel of the *internal candidates buffer* can represent candidates of more than one area as the extruded areas of the objects will typically overlap. Therefore, we represent the id of an area as one bit from the 128 bits available in the pixel of the *internal candidates buffer* as well as in the *id buffer*. In the following examples, we demonstrate the principle with 3 bits only as the remaining 125 bits are 0.

In Figure 3(a), we depict the internal label candidates for the configuration of three simple objects from Figure 2. Each pixel of the blue (id = 001_b), red (id = 010_b), and green (id = 100_b) regions represents one internal label candidate of Object A, Object B, and Object C, respectively. We depict label boxes of several internal label candidates for each region. In the violet region (id = $001_b \vee 010_b = 011_b$), the pixels represent internal label candidates of both Object A and Object B. The width and height of the label boxes of the candidates are given by the values in the list of the dimensions $\mathbb{D}$. Note that in the violet region, the dimensions of the label boxes of the internal label candidates of Object A are different from the dimensions of the label boxes of the internal label candidates of Object B.

To establish the external label candidates, we are using the approach of Čmolík and Bittner [25] modified to allow placement of external labels close to objects with non-convex shape. We have changed the definition of the internal area. We use the combined area of all objects, instead of the convex hull of the objects, as the internal area.

We define an external label candidate $\mathbf{c_E}$ as a triplet $\mathbf{c_E} = (\mathbf{a}, \boldsymbol{\pi}, \mathbf{l})$. The anchor $\mathbf{a}$ is a pixel of the area of the labeled object. The port $\boldsymbol{\pi}$ is the pixel located on the silhouette of the dilated internal area (dashed line in Figure 3(b)) that is closest to the anchor $\mathbf{a}$. The line connecting the anchor $\mathbf{a}$ and the port $\boldsymbol{\pi}$ defines the leader line of the external label candidate $\mathbf{c_E}$. The label box is connected to the port $\boldsymbol{\pi}$ in a corner point of the label box or a midpoint of one of its sides. We can determine the position of the label box from the angle α between the positive direction of the x-axis and the leader line pointing from the anchor $\mathbf{a}$ to the port $\boldsymbol{\pi}$. Based on the angle, the following corner of the label box is at the position of the port $\boldsymbol{\pi}$: bottom left corner for $\alpha \in (0°, 90°)$, bottom right corner for $\alpha \in [90°, 180°)$, top right corner for $\alpha \in (180°, 270°)$, and top left corner for $\alpha \in [270°, 360°)$. If the angle α is $0°$ or $180°$, then the midpoint of the left or right side of the label box is at the position of the port $\boldsymbol{\pi}$, respectively. From the label box position and its dimensions, it is straightforward to determine the position of the lower-left corner $\mathbf{l}$ of the label box. As both $\boldsymbol{\pi}$ and $\mathbf{l}$ depend on the position of the anchor $\mathbf{a}$, we represent each external label candidate $\mathbf{c_E}$ as a pixel of the *external candidates buffer* whose position corresponds to the position of the anchor $\mathbf{a}$.



(a) (b)

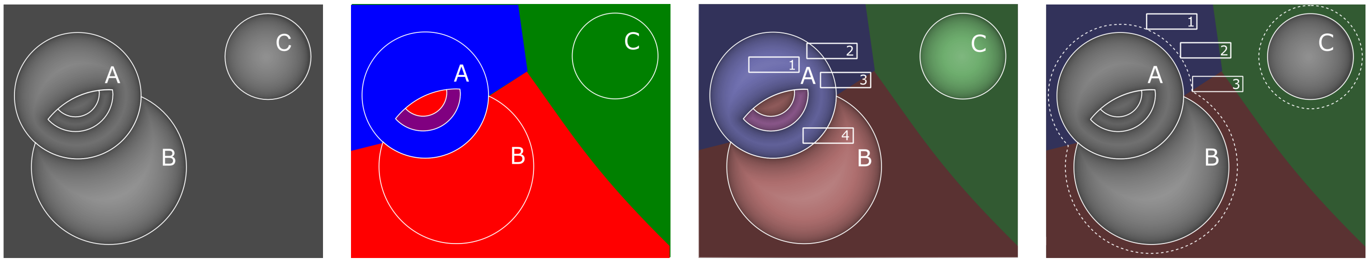
Fig. 3. (a) Internal label candidates obtained by dilating each area in the *id buffer* to the left by the width of the corresponding label and downwards by the height of the corresponding label. This way, the label box of each internal label candidate will overlap at least one pixel of the corresponding area in the *id buffer*. (b) External label candidates.

We can also restrict the directions of the leader lines (e.g., only to the left and right, only upwards and downwards). Without restricting the directions of the leader lines, the leader lines are perpendicular to the silhouette of the dilated internal area.

In Figure 3(b), we depict some external label candidates for the configuration of the three simple objects from Figure 2. Again, each pixel of the blue (id = 001_b), red (id = 010_b), and green (id = 100_b) regions represents one external label candidate of Object A, Object B, and Object C, respectively. The width and height of the label boxes of the external label candidates are again given by the values in the list of the dimensions $\mathbb{D}$. We depict the label boxes and leader lines of several external candidates. Similarly, as for the internal label candidates, in the violet region (id = $001_b \vee 010_b = 011_b$), the external label candidates represent candidates of both Object A and Object B. For the three simple objects, the internal area is the combined area of Object A, Object B, and Object C. In this case, the internal area is disconnected and non-convex. The dilated silhouette of the internal area is depicted with a dashed line.

We need to ensure that both the internal and the external labels are entirely inside of the *color buffer*. Otherwise, the labels would not be fully visible. Thus, we discard both the internal and the external label candidates whose label boxes are not entirely inside of the *color buffer*. We depict those label candidates in Figures 3(a) and 3(b) with a lighter color.

Further, we need to ensure that the external labels do not overlap the internal area heavily. Such overlaps are possible as the internal area can have a non-convex shape, see Figure 3(b), where label boxes of two external label



(a) Internal salience buffer

(b) Voronoi buffer

(c) Evaluation of internal label candidates

(d) Evaluation of external label candidates

Fig. 4. (a) The salience of the pixels in the *internal salience buffer* is computed as the distance to the closest point on the area outlines defined as discontinuities in the *id buffer*; lighter color means higher salience. The outlines are depicted in white color. (b) *Voronoi buffer* with regions color-coded based on the object ids. (c) Evaluation of label salience of four internal label candidates. (d) Evaluation of label salience of three external label candidates.

candidates of Object B (red and violet area) overlap Object C. We allow to control whether such overlaps are allowed and how big they can be with an overlap threshold t_o. We discard all external label candidates whose overlap of their label boxes with the internal area exceeds the threshold t_o. Both the overlap and the overlap threshold t_o are expressed in pixels (e.g., number of pixels of the internal area that the label boxes can overlap).

3.3 Labeling Criteria

To determine the positions of the labels, we evaluate each label candidate according to five criteria. To aggregate the criteria into the *fitness* F of the label candidate, we utilize Multiple Criteria Decision Making based on fuzzy logic [37]. We model each criterion C_i, $i \in \{1, \ldots, 5\}$ as a fuzzy membership function where we obtain a value in the range $[0, 1]$ for each label candidate. Further, we use weights W_i, $i \in \{1, \ldots, 5\}$ to control the strength of each criterion. To combine all the criteria together, we use non-compensating fuzzy aggregation, where one criterion cannot compensate for another criterion. More specifically, we use the natural T-norm that corresponds to standard multiplication. To aggregate all criteria for a label candidate into the *fitness* F of the candidate, we compute the product of all membership functions $C_i^{W_i}$ of the five criteria. In the following paragraphs, we describe the used criteria in detail.

Label Salience of Internal Label Candidates

To prioritize the unambiguous positions of the internal labels, we need to position each internal label into a central part of the area of the associated object. If the internal label does not fit entirely into the area of the associated object, then we need to minimize the overlap of the label with the areas of other objects, especially with their central parts. In such a case, we prefer overlap of the label with space outside of the internal area that is close to the associated object, but not close to areas of other objects.

To achieve this, we need to calculate the salience of each internal label candidate as an estimate of the ambiguity of the candidate. Higher salience corresponds to lower ambiguity. To do so, we utilize two additional buffers: an *internal salience buffer* and a *Voronoi buffer*. We create both these buffers by utilizing the information in the *id buffer*. The *internal salience buffer*, see Figure 4(a), stores in each pixel $\mathbf{p}$ its salience $S(\mathbf{p})$ calculated as

$$S(\mathbf{p}) = \begin{cases} s_I & \text{if } Id(\mathbf{p}) = 0, \\ (1 - s_I) \cdot \frac{dist(\mathbf{p},\mathbf{o})}{d_{max}} + s_I & \text{otherwise,} \end{cases} \quad (1)$$

where $dist(\mathbf{p}, \mathbf{o})$ is the distance from the pixel $\mathbf{p}$ to the closest pixel $\mathbf{o}$ on the outlines detected as discontinuities in the *id buffer*. Please note that for each pixel of the *id buffer*, the discontinuity is a binary value: 0 if the ids of all neighboring pixels equal the id of the pixel and 1 otherwise. Alternatively, we can see $dist(\mathbf{p}, \mathbf{o})$ as the radius of the largest circle with center at $\mathbf{p}$ inscribed in the corresponding area. When we establish external label candidates, we determine the length of their leader lines. We take d_{max} as the length of the longest leader line. Since $dist(\mathbf{p}, \mathbf{o}) \leq d_{max}$, we ensure that $S(\mathbf{p}) \in [0, 1]$. $Id(\mathbf{p})$ is the value stored in the *id buffer* at the position of pixel $\mathbf{p}$; if the value is 0, then no ids are stored at the position. Finally, $s_I \in [0, 1]$ is a user-defined parameter specifying the salience of pixels outside of the internal area.

The *Voronoi buffer* stores in each pixel $\mathbf{p}$ the id of the area whose outline is the closest to the pixel $\mathbf{p}$, see Figure 4(b). We use the *Voronoi buffer* as an estimate of the area the viewer will associate with a pixel on the screen. Moreover, we use the *internal salience buffer* as an estimate of the strength of this association. In Figure 4(c), the pixels are color-coded based on the ids in the *Voronoi buffer*. The lightness of the color indicates the salience of the pixels; lighter color corresponds to more salient pixels.

In Figure 4(c), we depict four possible placements of an internal label of Object A to illustrate how we can evaluate the salience of the internal label candidates based on the *internal salience buffer* and the *Voronoi buffer*. From the four depicted internal labels, we prefer Label 1 in the most central part of Object A as such an internal label can be very easily associated with Object A. If the internal label cannot be positioned entirely inside of Object A, then we prefer Label 2 that is not overlapping the red and green regions. Such an internal label can be again easily associated with Object A, as it is not near any other object. The remaining Labels 3 and 4, overlapping the red region, are not preferred as they reduce the space available for both internal and external labels of Object B. Further, Label 4 overlaps Object

B and thus is ambiguous as it can also be associated with Object B.

To achieve the unambiguous positioning of internal labels, we define two criteria to evaluate the salience of internal label candidates. The criteria are evaluated with respect to the area A_S selected for labeling.

The criterion C_1 evaluates the salience of the internal label candidate $\mathbf{c_I}$ with respect to the region R_S in the *Voronoi buffer* with the same id as the area A_S selected for labeling, (e.g., the blue region in Figure 4(c)) to favor the internal label candidates that overlap salient pixels in the area A_S as much as possible.

$$C_1(\mathbf{c_I}) = (1 - p_1) \cdot avg(\mathbf{c_I}, R_S) + p_1, \tag{2}$$

where $avg(\mathbf{c_I}, R_S)$ is the average salience of pixels in the region R_S and inside of the label box of the candidate $\mathbf{c_I}$. With the parameter $p_1 \in [0, 1]$, the user can increase the salience of label candidates of the area A_S selected for labeling. Note that $avg(\mathbf{c_I}, R_S) \in [0, 1]$, therefore $C_1(\mathbf{c_I}) \in [0, 1]$.

On the other hand, the criterion C_2 penalizes those internal label candidates that overlap with the set of regions $\mathbb{R}$ in the *Voronoi buffer* with an id different from the area A_S selected for labeling. The criterion is calculated as

$$C_2(\mathbf{c_I}) = \prod_{R \in \mathbb{R}} (1 - avg(\mathbf{c_I}, R)), \tag{3}$$

where $avg(\mathbf{c_I}, R)$ is the average salience of pixels in the region $R \in \mathbb{R}$ and inside of the label box of the candidate $\mathbf{c_I}$. Note that since $avg(\mathbf{c_I}, R) \in [0, 1]$, therefore $C_2(\mathbf{c_I}) \in [0, 1]$.

Label Salience of External Label Candidates

Similarly, as for internal labels, to prioritize the unambiguous positions of external labels, we need to position each external label next to the area of the associated object, but not close to areas of other objects. Further, we need the anchor of its leader line to be in a central region of the area of the associated object.

We use the same criteria C_1 and C_2 to evaluate the salience of each external label candidate. To evaluate the salience, we use an *external salience buffer* calculated with Equation 1 where we use $s_E \in [0, 1]$ instead of s_I.

Further, for the criterion C_2, we treat the internal area as an additional region to penalize overlap of the external label with its associated object in case the associated object has a non-convex shape and overlaps of external labels with the internal area are allowed. Figure 4(d) shows our example with three simple objects, where the pixels are color-coded based on the ids in the *Voronoi buffer* except for the internal area. The lightness of the color indicates the salience of pixels; lighter color corresponds to more salient pixels.

In Figure 4(d), we depict three possible placements of an external label of Object A to illustrate how we can evaluate the salience of the external label candidates based on the *external salience buffer* and the *Voronoi buffer*. From the three external labels, Label 1, which is entirely in the blue region is preferred the most. Note that such an external label can be very easily associated with Object A since it is not near any other object. Label 2, overlapping the green region, is less preferred as it reduces the space available for external labels of Object C. Label 3 overlaps the red and green regions and is not preferred as it can reduce the space available for external labels of both Objects B and C.

Anchor Salience of External Label Candidates

For each external label candidate, whose position is determined by the position of the anchor of its leader line, we further need to evaluate its salience. Again, the salience of each anchor is an estimate of the ambiguity of the anchor. Higher salience corresponds to lower ambiguity. We calculate the salience with the approach of Čmolík and Bittner [7] as

$$C_3(\mathbf{c_E}) = \frac{dist(\mathbf{a}, \mathbf{o})}{d_{max}}, \tag{4}$$

where $\mathbf{a}$ is the position of the anchor of the external label candidate $\mathbf{c_E}$, $\mathbf{o}$ is position of the closest point to anchor $\mathbf{a}$ on the outlines detected as discontinuities in the *id buffer*, d_{max} is the maximum length of the leader line of all external label candidates, and $dist$ gives us the distance between the two points. Note that $dist(\mathbf{a}, \mathbf{o}) \le d_{max}$, which means that $C_3(\mathbf{c_E}) \in [0, 1]$. Further, note that the distance is stored in the *external salience buffer* at the position of the anchor $\mathbf{a}$ of the external label candidate $\mathbf{c_E}$. For internal label candidates the criterion C_3 is always 1 as they do not have anchors.

Leader Line Length

The leader lines of the external labels should be as short as possible, while still pointing to the central part of the area of the associated object. Therefore, we evaluate the length of the leader line for each external label candidate with the approach of Čmolík and Bittner [7] as

$$C_4(\mathbf{c_E}) = 1 - \frac{dist(\mathbf{a}, \boldsymbol{\pi})}{d_{max}}, \tag{5}$$

where $dist(\mathbf{a}, \boldsymbol{\pi})$ is the distance between the position of the anchor $\mathbf{a}$ and the position of the port $\boldsymbol{\pi}$ of the external label candidate $\mathbf{c_E}$, and d_{max} is the length of the longest leader line. Note that $dist(\mathbf{a}, \boldsymbol{\pi}) \le d_{max}$ and, therefore, $C_4(\mathbf{c_E}) \in [0, 1]$. As internal label candidates do not have leader lines, the criterion C_4 is always 1 for internal label candidates.

Area Ambiguity

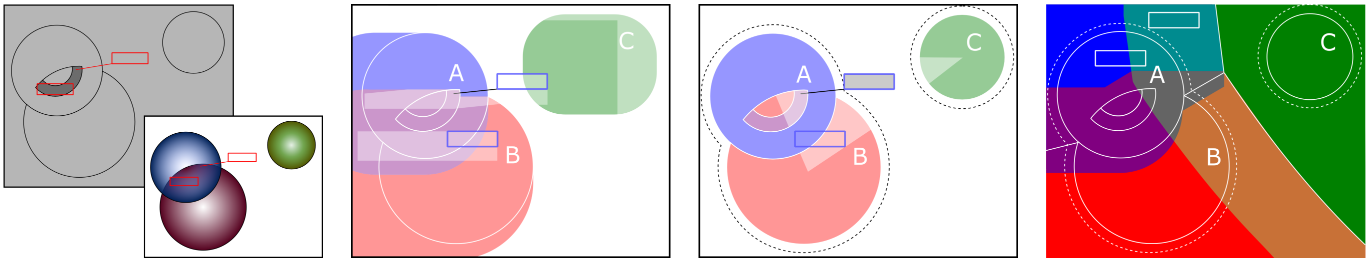
In case that the labeled objects are semi-transparent, we prefer positioning the internal labels or anchors of the external labels to regions where the areas of the objects are overlapping as little as possible. Otherwise, the association of the labels to the labeled objects can be ambiguous. To evaluate the area overlaps, we calculate the number of ids in each pixel of the *id buffer* and store it in the *count buffer*. For pixels outside of the internal area, we put 1 in the *count buffer*. Otherwise, we would prefer internal label candidates that are overlapping the internal area as little as possible. Figure 5(a) shows the *count buffer* of the example with three simple objects and ambiguous positions of the labels.

To evaluate internal label candidates, we use the criterion

$$C_5(\mathbf{c_I}) = 1 - \frac{k}{m}, \tag{6}$$

where k is the average number of areas for all pixels in the label box of $\mathbf{c_I}$ and m is the maximum number of areas. To efficiently obtain the average number of areas, we calculate the Summed Area Table of the *count buffer*.

Similarly, to evaluate the external label candidates, we use again the approach of Čmolík and Bittner [7] that results in the same equation, but k is the number of areas in the pixel at the anchor position of the external label candidate.



(a) Count buffer (b) Eliminated internal label candidates (c) Eliminated external label candidates (d) Lookup buffer

Fig. 5. (a) *Count buffer* with an example of the ambiguous placement of labels. (b) Eliminated internal label candidates and (c) eliminated external label candidates, depicted in a light color, after placement of one internal and one external label. (d) *Lookup buffer* created from the *Voronoi buffer*. We need to look up one tile for the blue, red, and green regions. Two tiles for the cyan, violet, and brown regions. And tree tiles for the grey region.

3.4 Establishing Buffers for the Labeling Criteria

As a preprocessing step, we evaluate the *fitness* of all internal and external label candidates by the criteria C_3, C_4, and C_5 only. We store the *fitness* of the internal label candidates in the *internal fitness buffer*. Similarly, we store the *fitness* of the external label candidates in the *external fitness buffer*.

Further, we create the *internal salience buffer* and *Voronoi buffer* in this preprocessing step. However, we evaluate the internal and external label candidates according to the criteria C_1 and C_2 using these two buffers later, when we search for the best internal and best external label candidate.

3.5 Selecting an Area for Labeling

The order in which the labels are placed over the illustration is crucial as our method is based on a greedy algorithm, and we cannot recover from a bad partial solution, i.e., a state when some unlabeled area has no further label candidates. We could use a genetic algorithm, as in [29], to alter the order in which we position the labels in case of a bad partial solution. However, such an approach would result in higher computation times and, in turn, the algorithm would not operate in real-time.

Placing a label for one area over the illustration can reduce the number of available label candidates of the other areas. Therefore, we should label the areas with a low number of good label candidates first.

To select one of the unlabeled areas as the area A_S for labeling, we calculate the *capacity* of each area as the sum of the salience of all internal label candidates of the area and choose the unlabeled area for which the *capacity* is the lowest. We use the salience of internal label candidates to calculate the *capacity* as we try to label the objects with internal labels first. To evaluate the salience of each internal label candidate, we use only the criterion C_1. Note that we need to recalculate the *capacities* of the areas each time that we place a new label over the illustration. Further, when we place the label for the selected area A_S over the illustration, we need to mark the selected area A_S as labeled.

3.6 Finding the Best Label Candidate

To find the best internal label candidate $\mathbf{c_I}$ for the selected area A_S, we need to find the internal label candidate with the maximum *fitness* F. To do so, we evaluate all internal label candidates of the area A_S by the criteria C_1 and C_2. We calculate the *fitness* F of each candidate by multiplying the value stored in the *internal fitness buffer* with C_1 and C_2.

If the *fitness* of the best internal label candidate $\mathbf{c_I}$ is lower than the user-specified ambiguity threshold t_a, then we need to find the best external label candidate $\mathbf{c_E}$ with the maximum *fitness* F. Similarly, as for the internal label candidates, we evaluate all external label candidates of the area A_S according to the criteria C_1 and C_2 and calculate the *fitness* F of each candidate by multiplying the value stored in the *external fitness buffer* with C_1 and C_2.

At this point, we do not compare the quality of the best external label candidate with the quality of the best internal label candidate and always use the best external label candidate. Further research in this direction is required. Only if there are no external label candidates, then we use the best internal label candidate $\mathbf{c_I}$ with *fitness* F below the ambiguity threshold t_a.

3.7 Eliminating Overlapping Label Candidates

We need to ensure that the placed labels do not overlap with each other. For the external labels, we further need to ensure that their leader lines do not overlap with the internal labels.

To prevent such overlaps, we simply update the *internal label candidates buffer* and the *external label candidates buffer* and discard those label candidates that overlap with the new label determined for the selected area A_S. If the label determined for the selected area is positioned externally, then we also discard label candidates in the *internal label candidates buffer* that overlap with the leader line of the determined external label.

In Figure 5(b), we depict the internal label candidates discarded after one internal label and one external label are determined. Similarly, in Figure 5(c), we depict the external label candidates discarded after one internal label and one external label are determined. In both figures, the discarded label candidates are indicated with a lighter color.

3.8 Implementation Details

In this section, we present the technical details related to the implementation of the first two steps of the proposed method. For a graphical overview with all used buffers, please refer to the supplementary material.

We establish the internal label candidates by dilating each area A_i in the *id buffer* to the left by the width w_i of the label and down by the height h_i of the label with Jump flooding [38]. To establish the external label candidates, we are using the approach of Čmolík and Bittner [25] adapted to support non-convex shapes of internal areas.

To eliminate external label candidates that overlap the internal area, we create an image with a black background and the internal area in white color. We calculate the Summed Area Table [39] of the image that allows us to obtain the sum of values of pixels (i.e., the number of white pixels in this case) in every rectangle in the image with just four texture lookups. Using the Summed Area Table of the image, we obtain the number of white pixels inside of the label box of each external label candidate and discard the external label candidates for which the number of pixels inside of the internal area is larger than the given overlap threshold t_o.

We are using Jump flooding to create the *internal salience buffer* and *Voronoi buffer* by calculating the Voronoi diagram of the outlines detected as discontinuities in the *id buffer*.

We use scattering [40] to efficiently find both the internal and external label candidates with the maximum *fitness* F. Further, to efficiently evaluate the criteria C_1 and C_2, in particular, the average salience of pixels inside of the label box of each internal label candidate with respect to the regions in the *Voronoi buffer*, we distribute the *internal salience buffer* into tiles of the *internal tile buffer* using the ids in the *Voronoi buffer* such that each tile of the *internal tile buffer* contains only the salience of pixels in one region of the *Voronoi buffer*. E.g., each tile will contain only one of the three color-coded regions in Figure 4(c). The violet region will be both in the tile of the blue area and in the tile of the red area. Then, we calculate the Summed Area Table of each tile that allows us to obtain the sum in every rectangle in the tile with just four texture lookups.

To further speed up the calculation of criterion C_2, we dilate the cells of the Voronoi diagram in the *Voronoi buffer* to the left by the maximum width of all labels w_{max} and down by the maximum height of all labels h_{max} and store them in the *lookup buffer*. We use the ids in the *lookup buffer* at the position of an internal label candidate to reduce the number of tiles, which we need to look up to evaluate criterion C_2 for the candidate. Figure 5(d) shows an example of the lookup buffer with one internal label candidate. For the internal label inside of Object A, we need to look up only the blue tile of the *internal tile buffer* as the label cannot overlap any other region in the *Voronoi buffer*.

Similarly, as for the internal label candidates, we distribute the *external salience buffer* into tiles of an *external tile buffer* using the ids in the *Voronoi buffer* and the *id buffer*. Each tile of the *external tile buffer* contains only the salience of pixels in one region of the *Voronoi buffer* that are outside of the internal area, and we add one tile for the internal area. E.g., each tile will contain only one of the three color-coded regions in Figure 4(d). The fourth tile for the internal area will contain the grey region. Then, we calculate the Summed Area Table of each tile and use the *lookup buffer* to reduce the number of lookups needed to evaluate criterion C_2. We always look up the tile of the internal area. Figure 5(d) shows an example of the lookup buffer with the label box of one external label candidate. For the external label candidate, we need to look up the blue, green, and internal area tiles of the *external tile buffer* based on the position $\mathbf{l}$ of its label box.

4 Results

We evaluated the proposed method with implementation in Java and OpenGL. For all label layouts presented in this paper, the supplementary material, and the supplementary video, we used the same parameters $s_I = 0.1$, $s_E = 0.1$, $p_1 = 0.1$, overlap threshold $t_o = 0$, and weights of the criteria $W_1 = 1$, $W_2 = 5$, $W_3 = 1$, $W_4 = 1$, $W_5 = 5$ except for the 3D model of a head where we used the weights $W_1 = 1$, $W_2 = 1$, $W_3 = 1$, $W_4 = 1$, $W_5 = 0.5$ due to heavy overlaps of the objects. The only other parameter that is varying for the presented label layouts is the ambiguity threshold t_a. The values of the weights, parameters, and thresholds were selected as values for which most of the label layouts looked the best after experimenting with various values.

The main contribution of the proposed method is the ability to place both internal and external labels over the illustration, while the internal labels can also partially overlap the labeled object, no two labels overlap, and no label is intersecting the leader lines of the external labels.

We demonstrate the benefits of using internal labels that only partially overlap the labeled objects on the example of the Gapminder dataset [41]. If we use only labels that are fully enclosed in their corresponding objects (Figure 6(a)), then we can label only four objects. If we add external labels to the labels that are fully enclosed in their corresponding objects (Figure 6(b)), then we can label most of the objects, but five of the objects still remain unlabeled. When we allow the internal labels to overlap their corresponding objects only partially (Figure 6(c)), then we are able to label all objects. Note that all objects except two are labeled with internal labels. Another possibility is to increase the ambiguity threshold t_a to use labels partially overlapping areas of the corresponding objects only for the unlabeled objects in Figure 6(b). Please see Figure 6(d) for the result and refer to the supplementary material for more examples.

As we can see, the user can set the ambiguity threshold t_a to control the allowed ambiguity of the internal labels and force the method to use external labels instead of the ambiguous internal labels. Figure 1 shows another example.

By changing the ambiguity threshold t_a, we are able to produce label layout styles for area features in cartography (Figure 10(a)) and for data visualizations (Figure 6) where partial overlaps of internal labels are allowed, for medical illustrations where internal labels are typically inside of the labeled objects (Figure 9(a)), and for technical illustrations where the objects are labeled externally (Figure 7(d)).

We can use the proposed method with various directions of the leader lines of external labels. In Figure 7, we show several mixed label layouts with various directions of leader lines of external labels.

The proposed method is able to position external labels around a non-convex internal area, which allows the use of the label layout even when we zoom in close to the labeled objects. We demonstrate this ability in Figure 7(e).

Further, the proposed method is able to position labels over renderings of semi-transparent objects. We have used an extended approach of Kruger et al. [42] to render the

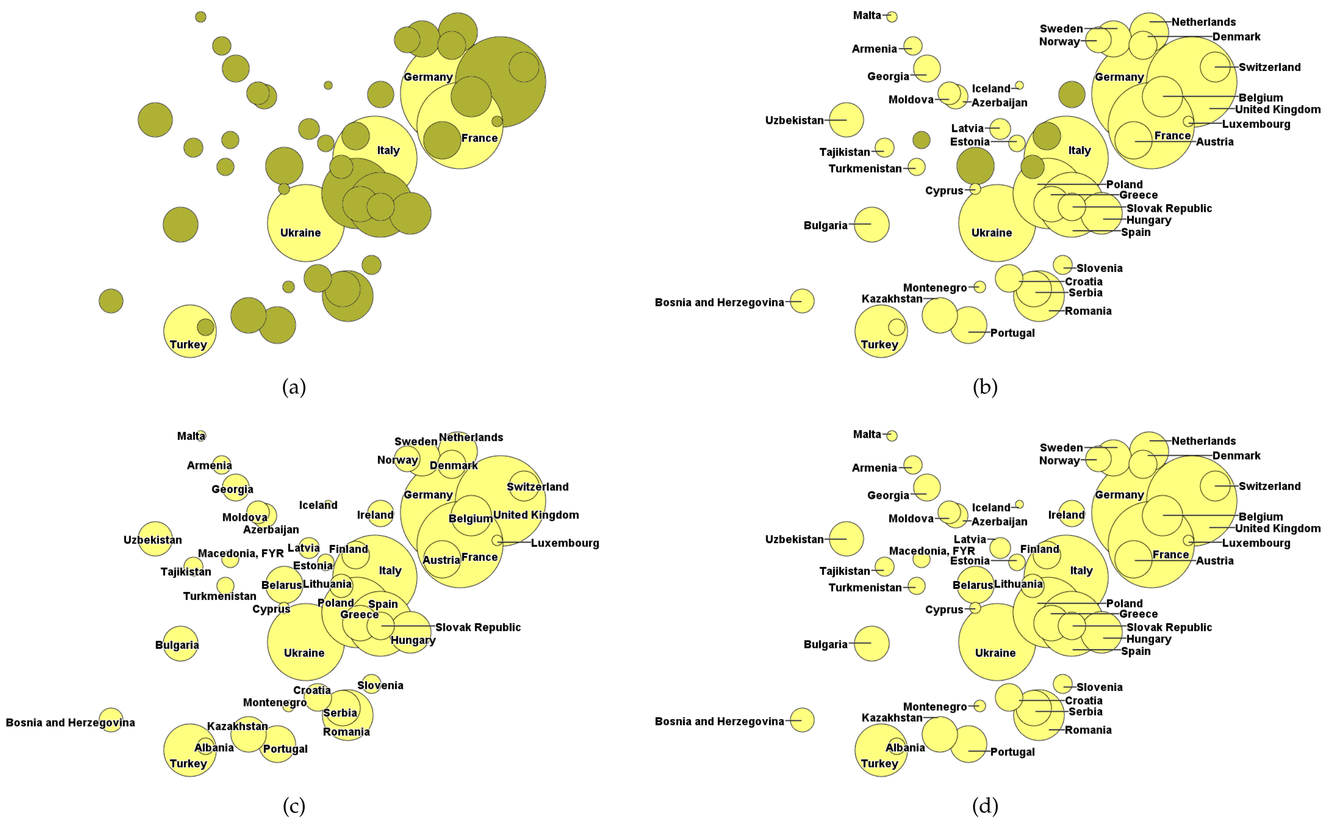


Fig. 6. Visualizations of the Gapminder data set labeled with various combinations of label types: (a) Only labels fully contained inside of their corresponding areas. The unlabeled areas are highlighted in a darker color. (b) Labels fully contained inside of their corresponding areas together with external labels. The unlabeled areas are highlighted in a darker color. (c) Labels fully contained inside of their corresponding areas together with labels partially overlapping their corresponding areas and few external labels. (d) Labels fully contained inside of their corresponding areas together with external labels and few labels partially overlapping their corresponding areas.

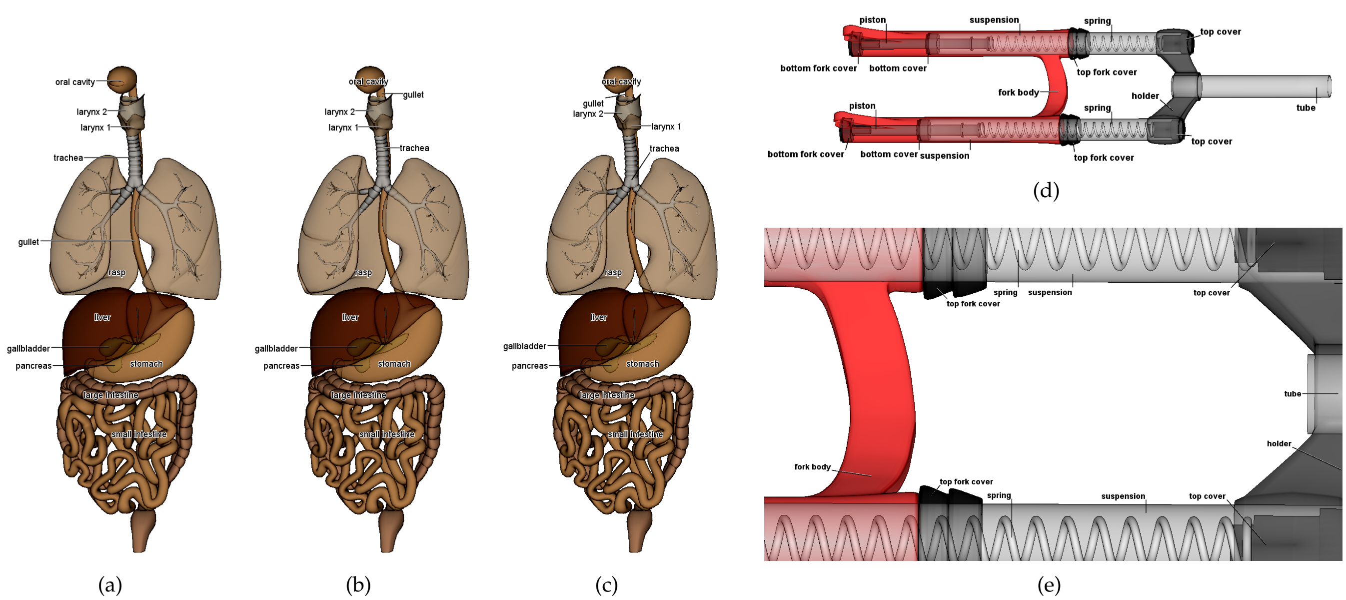


Fig. 7. Label layouts with various directions of the leader lines: (a) to the left only, (b) to the left and to the right, (c) all directions. (d) Technical illustration using only external labels. (e) An example of zooming into a non-convex region of the internal area.

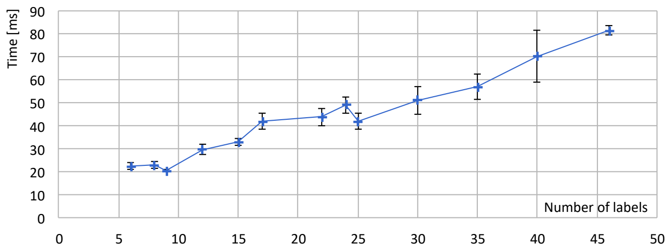


Fig. 8. Average time needed to calculate the mixed label layout in dependency on the number of labeled objects. The lower error bars represent the time needed to calculate the label layout with all labels positioned internally. The upper error bars represent the time needed to calculate the label layout with all labels positioned externally.

semi-transparent objects. In Figures 1 and 7, we utilize the proposed method to label semi-transparent objects.

Nevertheless, we can use the proposed method with any algorithm capable of producing the *color buffer* and *id buffer*. To demonstrate this ability, we have created an application that produces the *color buffer* and *id buffer* for the Gapminder dataset, see Figure 6. Further, we have created an application that is able to load images of the *color buffer* and *id buffer*. We have used the application to create label layouts for a handmade illustration, see Figure 9(c), and a map of the Caribbean, see Figure 10(a).

The asymptotic computational complexity of the proposed method is $O(n^2)$ as the method sequentially determines positions of n labels, and to determine the position of each label, it needs to look up $O(n)$ tiles (to calculate the criterion C_2) in the worst case. However, with the *lookup buffer*, the method needs to look up only $O(1)$ tiles for most configurations of the objects. Therefore, for most configurations of the objects, the computational complexity will be $O(n)$, which matches the measured performance of the proposed method in dependency on the number of labeled objects depicted in Figure 8.

For the performance measurements, we have used a PC running Windows 10 with 64 GB of DDR4 RAM, Intel Xeon W-2125 CPU with 4 cores running at 4 GHz, and NVIDIA TITAN Xp GPU equipped with 12 GB of GDDR5X RAM, 3840 unified shaders, and 240 texture units. We have used scenes with 6 to 46 objects to be labeled with 1024×1024 *color buffer*. Resolution of all other buffers and each tile of the tile buffers was 512×512. For all tested scenes, the proposed method calculates the label layout in under 100 ms. In other words, according to the classification of response times by Nielsen [43, Section 5.5], the proposed method gives the results immediately. The supplementary video shows a live capture of the prototype application.

5 Limitations

The proposed method has several limitations. We give examples for a selected subset of them in the supplementary material. While it is able to position external labels around a non-convex internal area, in certain cases, a large number of external label candidates will point their leader lines to the same location. In such a case some of the objects cannot be labeled externally as there is no room for all labels of the objects. Still, such objects will be labeled internally and potentially ambiguously in the proposed method.

Similar issues will arise if we restrict the direction of leader lines to the vertical direction only. Then, the external labels, especially longer labels, will occupy all the free space for external labels, and some of the objects will not be labeled externally. Again, such objects will be labeled internally and potentially ambiguously in the proposed method.

For rare configurations of objects, the algorithm can discard all label candidates of a certain object before the object is labeled. In such a case, the algorithm will yield a solution, where the object is not labeled. In other words, it is not guaranteed that the algorithm will always find a solution where all objects are labeled. However, we have not experienced such a case in our experiments. To resolve such situations, the labels could be replaced with shorter references (numbers, letters, or abbreviations) to a legend containing the full labels.

The proposed method does not take into account the semantics of the labeled objects, which could influence what parts of the objects are more or less important. A simple solution might be to let the user mark semantically important regions on the 2D or 3D model and use this information when calculating the salience of label candidates.

The proposed method is able to work with one-line labels that are aligned with the horizontal axis only. In the future, we would like to extend our method to support multi-line labels and labels not aligned with the horizontal axis that are utilized for labeling of long and thin area features in the approach of Götzelmann et al. [9].

The proposed method does not make the movement of the labels temporally coherent, and the labels may jump abruptly during interaction with the scene, especially with a 3D scene. Therefore, we hide and do not calculate the label layout during the interaction. We have tried to incorporate the criteria for temporal coherence of Čmolík and Bittner [25] but did not achieve temporally coherent movement of the labels. Due to semi-transparent objects, there are many more discontinuities in the *internal salience buffer* and *external salience buffer*. Further, in our approach, the labels can change their type from internal to external and vice versa. We would like to combine our approach with the approaches of Tatzgern et al. [44] and Kouřil et al. [6] to label 3D scenes during interaction in the future.

6 Expert Evaluation

To assess the feasibility of the proposed method, we have conducted an expert evaluation with an infographics illustrator. The main interests of the evaluation were to what extent the proposed method can satisfy a professional illustrator and what are the essential factors for a good label layout from a professional point of view.

We invited a professional illustrator with over five years of experience in infographics design. She mainly works on signage and guidance diagrams for visitors inside buildings, and most of her work includes labeling tasks.

In the evaluation, we have used three datasets: a 3D model of a human head, an illustration of the Zika virus, and a map of the Caribbean. We asked the illustrator to perform two tasks on each dataset: (1) Design an appropriate label placement for the dataset, and (2) evaluate the result of the

(a) (b) (c) (d)

Fig. 9. The label layout calculated with the proposed method for the 3D model of a head (a) and the label layout created manually by a professional illustrator (b). The label layout calculated with the proposed method for the illustration of the Zika virus by David S. Goodsell (CC-BY-4.0) with the id buffer (c), and the label layout created by the professional illustrator (d). Note that the illustrator accidentally switched the labels for RNA and Capsid proteins. In order to have better readability in the paper, the leader lines drawn by the illustrator are thickened through image editing in (b) and (d).

(a) (b)

Fig. 10. The label layout calculated with the proposed method for the map of the Caribbean with the id buffer (a) and the label layout created manually by the professional illustrator (b). Note that we accidentally misspelled Guadeloupe as Guadaloupe in the system. We keep the typo here to have a fair comparison with the result created by the illustrator.

proposed method (which was created before the evaluation) and point out and explain any insufficiencies.

For each dataset, we provided the illustrator with a background image together with the corresponding labels printed on transparent film cut into several small pieces. For each labeled object, the illustrator was allowed to place the label internally or externally with a straight leader line based on her preference. However, all labels had to be fully embedded within the image domain. Additional instructions regarding the context of the labels were provided and explained on demand.

Figures 9(b), 9(d), and 10(b) show the label layouts created by the illustrator. She created each label layout in 10 to 15 minutes. In the following, we describe rules that we have obtained directly from the illustrator (e.g., she explained to us that she is using such rules). Regarding the *Global Strategies*, three ideas are often incorporated. These include (G1) to first place internal labels, and then external labels, (G2) if possible, labels should not overlap objects that are also labeled, and (G3) identify regions without important features for the label placement. As rules for the *Internal Labels*, (I1) internal labels are often placed in the most central part of the objects. (I2) If the object is too small to accommodate the entire internal label, then place the left side of the internal label inside of the object. If that is not possible, then place the right side of the internal label inside of the object. If that is also not possible, place the center of the internal label inside of the object. As rules for the *External Labels*, (E1) a leader line is added to the target object for labels that are overlapping with other objects. (E2) If possible, the external labels are positioned with leader lines such that the leader lines are short.

Once the label layouts were finished, we showed her the results generated using our implemented method and asked for comments from a professional perspective. She was impressed by the results, especially being created by an algorithm, but she also pointed out labels violating her above-mentioned labeling rules in each result.

The 3D model of a head was considered by her as a simple scene to embed all labels as internal labels. She placed as many internal labels as possible, but for small and overlapping objects, she added a leader line to specify the exact object to be labeled. In particular, she connected the *pituitary* and *spinal cord* labels with the target objects with leader lines, since the target objects are small and overlapping with *temporal lobe* and *spine*, respectively. However, she positioned these labels on top of another object and not on the background to avoid long leader lines.

She placed the *skin* label outside of the head contour to point out that the skin is a container object covering the entire head. In the result of the proposed method, most of the internal labels are on positions close to those chosen by the illustrator. However, the external labels are positioned outside of the internal area and connected with the objects with long leader lines (violation of rule E2). Further, the label for skin is positioned as an internal label as the proposed method cannot derive contextual information such as the skin being a tissue covering the whole head.

The illustrator considered the Zika virus as a complex scene composed of several repetitive structures. To her, the only structure big enough to accommodate an internal label was the Zika virus itself. She suggested to adjust color contrast or add semi-transparent background boxes to differentiate the background image and text labels. One interesting property that she mentioned for this dataset is to add many-to-one leader lines to indicate multiple instances with the same semantic. Again, the leader lines in the result of our method are longer than in the label layout created by the illustrator (violation of rule E2). The illustrator put the label *Envelope proteins* such that it overlaps the Alpha-helix protein (the green branching structure) to emphasize Envelope proteins in both depicted Zika viruses.

For the map of the Caribbean, the illustrator placed the labels inside of the islands (rule G1) and determined their position based on how precise the labels can describe the region. Since most of the islands are round, placing the label over them is not an issue. However, some islands, such as *Guadeloupe*, have a characteristic shape, and therefore should not be covered by the label. If possible, she positioned the left or right side of the internal labels inside of the small islands. She did not find it necessary to use leader lines for this data set unless she would need to highlight a specific island. The centers of most labels are positioned inside of the small islands (violation of rule I2) in the result of our proposed method. Further, our method does not distinguish between round islands and islands with a characteristic shape. She liked the result for the map of the Caribbean the best among the three automated results overall.

In general, the proposed method positions the labels at similar locations as the illustrator. In future work, we aim to resolve when to apply rule I2 and when to position the label externally. Further, we need to allow external labels to be positioned over other objects (rule E1) and resolve when such placement should be preferred over positioning of an external label over the background. We believe that both these problems are highly dependent on the context and designers' preferences, and therefore require more sophisticated algorithms.

7 Conclusions and Future Work

In this paper, we have presented a method capable of mixed labeling of 2D and 3D objects, where the objects are labeled with both internal labels placed over (parts of) the objects and external labels placed in the space around the objects and connected with the labeled objects with leader lines. The presented method determines the position and type of each label based on the user-specified ambiguity threshold and eliminates overlaps between the labels and between the internal labels and the leader lines of external labels. The method is a screen-space technique that takes two images, where the 2D objects or projected 3D objects are encoded as the input. In other words, we can use the algorithm whenever we can render the objects as an image, which makes the algorithm fit for use in many domains. The method operates in real-time, giving the results immediately. We have presented the results of the proposed method to a professional illustrator and asked her to evaluate the label layouts produced with the proposed algorithm. The feedback from the illustrator was very positive. However, she pointed out one rule for the internal labels and one rule for the external labels that the proposed method is not yet considering. In the future, we would like to address the limitations of the proposed method and add the missing rules pointed out by the professional illustrator during the expert evaluation.

Acknowledgments

This research has been supported by MEYS of Czechia OP VVV grant No. CZ.02.1.01/0.0/0.0/16_019/0000765 – Research Center for Informatics, Grant Agency of CTU in Prague grant No. SGS19/179/OHK3/3T/13, EU Horizon 2020 MSCA grant No. 747985, and the Austrian Science Fund (FWF) grant P 31119.

References

[1] E. R. Tufte, *The visual display of quantitative information*, 2nd ed. Graphics Press, 2001.

[2] K. Ali, K. Hartmann, and T. Strothotte, "Label layout for interactive 3D illustrations," *Journal of the WSCG*, vol. 13, no. 1, pp. 1–8, 2005.

[3] C. Richards, "Technical and scientific illustration," in *Information Design: Research and Practice*. Taylor & Francis, 2017, ch. 5, pp. 85–106.

[4] S. Oeltze-Jafra and B. Preim, "Survey of labeling techniques in medical visualizations," in *VCBM'14*. Eurographics Association, 2014, pp. 199–208.

[5] P. Yoeli, "The logic of automated map lettering," *The Cartographic Journal*, vol. 9, no. 2, pp. 99–108, 1972.

[6] D. Kouřil, L. Čmolík, B. Kozlíková, H. 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 TVCG*, vol. 25, no. 1, pp. 977–986, 2019.

[7] L. Čmolík and J. Bittner, "Real-time external labeling of ghosted views," *IEEE TVCG*, vol. 25, no. 7, pp. 2458–2470, 2019.

[8] K. Been, M. Nöllenburg, S.-H. Poon, and A. Wolff, "Optimizing active ranges for consistent dynamic map labeling," *Computational Geometry: Theory and Applications*, vol. 43, no. 3, pp. 312–328, 2010.

[9] T. Götzelmann, K. Hartmann, and T. Strothotte, "Agent-based annotation of interactive 3D visualizations," in *Smart Graphics*, ser. LNCS, vol. 6543. Springer, 2006, pp. 24–35.

[10] J. W. van Roessel, "An algorithm for locating candidate labeling boxes within a polygon," *The American Cartographer*, vol. 16, no. 3, pp. 201–209, 1989.

[11] M. Barrault, "A methodology for placement and evaluation of area map labels," *Computers, Environment and Urban Systems*, vol. 25, no. 1, pp. 33–52, 2001.

[12] H. Freeman, "Automated cartographic text placement," *Pattern Recognition Letters*, vol. 26, no. 3, pp. 287–297, 2005.

[13] T. Ropinski, J.-S. Praßni, J. Roters, and K. Hinrichs, "Internal labels as shape cues for medical illustration," in *Vision, Modeling, and Visualization (VMV)*, 2007, pp. 203–212.

[14] G. Cipriano and M. Gleicher, "Text scaffolds for effective surface labeling," *IEEE TVCG*, vol. 14, no. 6, pp. 1675–1682, 2008.

[15] R. D. Prado and A. B. Raposo, "Real-time label visualization in massive CAD models," in *Int. Conf. Computer-Aided Design and Computer Graphics*. IEEE, 2013, pp. 337–344.

[16] S. Maass and J. Döllner, "Dynamic annotation of interactive environments using object-integrated billboards," in *Int. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision*. Pilsen, Czech Republic: Václav Skala-UNION Agency, 2006, pp. 327–334.

[17] M. A. Bekos, B. Niedermann, and M. Nöllenburg, "External labeling techniques: A taxonomy and survey," *Computer Graphics Forum*, vol. 38, no. 3, pp. 833–860, 2019.

[18] M. Rylov and A. Reimer, "A practical algorithm for the external annotation of area features," *Cartographic Journal*, vol. 54, no. 1, pp. 61–76, 2017.

[19] M. Nöllenburg, V. Polishchuk, and M. Sysikaski, "Dynamic one-sided boundary labeling," in *SIGSPATIAL Int. Conf. Advances in Geographic Information Systems*. ACM, 2010, pp. 310–319.

[20] Z.-D. Huang, S.-H. Poon, and C.-C. Lin, "Boundary labeling with flexible label positions," in *Algorithms and Computation (WALCOM)*, ser. LNCS, vol. 8344. Springer, 2014, pp. 44–55.

[21] B. Preim, A. Ritter, T. Strothotte, T. Pohle, D. R. Forsey, and L. Bartram, "Consistency of rendered images and their textual labels," in *Proc. of Edu + CompuGraphics*. Queluz, Portugal: GRASP, 1995, pp. 201–210.

[22] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis, "Area-feature boundary labeling," *The Computer Journal*, vol. 53, no. 6, pp. 827–841, 2010.

[23] M. Bekos, M. Kaufmann, D. Papadopoulos, and A. Symvonis, "Combining traditional map labeling with boundary labeling," in *Current Trends in Theory and Practice of Computer Science (SOFSEM)*, ser. LNCS, vol. 6543. Springer, 2011, pp. 111–122.

[24] M. Löffler, M. Nöllenburg, and F. Staals, "Mixed map labeling," *Journal of Spatial Information Science*, vol. 13, pp. 3–32, 2016.

[25] L. Čmolík and J. Bittner, "Layout-aware optimization for interactive labeling of 3D models," *Computers & Graphics*, vol. 34, no. 4, pp. 378–387, 2010.

[26] B. Niedermann, M. Nöllenburg, and I. Rutter, "Radial contour labeling with straight leaders," in *IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 2017, pp. 295–304.

[27] J.-D. Fekete and C. Plaisant, "Excentric labeling: Dynamic neighborhood labeling for data visualizaition," in *Proc. of CHI'99*. ACM, 1999, pp. 512–519.

[28] T. Stein and X. Décoret, "Dynamic label placement for improved interactive exploration," in *Int. Symp. Non-photorealistic Animation and Rendering (NPAR)*. ACM, 2008, pp. 15–21.

[29] H. Wu, S. Takahashi, C. Lin, and H. Yen, "A zone-based approach for placing annotation labels on metro maps," in *Smart Graphics*, ser. LNCS, vol. 6815. Springer, 2011, pp. 91–102.

[30] S. Maass and J. Döllner, "Efficient view management for dynamic annotation placement in virtual landscapes," in *Smart Graphics*, ser. LNCS, vol. 4073. Springer, 2006, pp. 1–12.

[31] A. Gemsa, J.-H. Haunert, and M. Nöllenburg, "Multi-row boundary-labeling algorithms for panorama images," *ACM TSAS*, vol. 1, no. 1, pp. 289–298, 2014.

[32] B. Bell, S. Feiner, and T. Höllerer, "View management for virtual and augmented reality," in *ACM Symp. User Interface Software and Technology (UIST)*. ACM, 2001, pp. 101–110.

[33] T. Götzelmann, K. Ali, K. Hartmann, and T. Strothotte, "Form follows function: Aesthetic interactive labels," in *Eurographics Conf. Computational Aesthetics in Graphics, Visualization and Imaging*. Eurographics Association, 2005, pp. 193–200.

[34] T. Götzelmann, K. Ali, K. Hartmann, and T. Strothotte, "Adaptive labeling for illustrations," in *Proc. of Pacific Graphics*, vol. 2005, 2005, pp. 64–66.

[35] M. Luboschik, H. Schumann, and H. Cords, "Particle-based labeling: Fast point-feature labeling without obscuring other visual features," *IEEE TVCG*, vol. 14, no. 6, pp. 1237–1244, 2008.

[36] L. Barth, A. Gemsa, B. Niedermann, and M. Nöllenburg, "On the readability of leaders in boundary labeling," *Information Visualization*, vol. 18, no. 1, pp. 110–132, 2019.

[37] R. E. Bellman and L. A. Zadeh, "Decision-making in a fuzzy environment," *Management Science*, vol. 17, no. 4, pp. B–141–164, 1970.

[38] G. Rong and T.-S. Tan, "Jump flooding in GPU with applications to Voronoi diagram and distance transform," in *Symp. Interactive 3D Graphics and Games (I3D)*. ACM, 2006, pp. 109–116.

[39] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra, "Fast summed-area table generation and its applications," *Computer Graphics Forum*, vol. 24, no. 3, pp. 547–555, 2005.

[40] T. Scheuermann and J. Hensley, "Efficient histogram generation using scattering on GPUs," in *Symp. Interactive 3D Graphics and Games (I3D)*. ACM, 2007, pp. 33–37.

[41] Gapminder Foundation, https://www.gapminder.org/data/. [Online; accessed 19-June-2019].

[42] J. Krüger, J. Schneider, and R. Westermann, "Clearview: An interactive context preserving hotspot visualization technique," *IEEE TVCG*, vol. 12, no. 5, pp. 941–948, 2006.

[43] J. Nielsen, *Usability engineering*. Elsevier, 1994.

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

Ladislav Čmolík is an assistant professor at the Department of Computer Graphics and Interaction of the Czech Technical University in Prague, Czechia. He received his PhD from the same institution in 2011. His research interests include illustrative visualization, non-photorealistic rendering, and HCI.

Václav Pavlovec is a PhD student at the Czech Technical University in Prague, Czechia. He received his master degree from the same institution in 2019. His research interests include illustrative visualization and HCI.

Hsiang-Yun Wu is a Postdoctoral Research Fellow at the Institute of Visual Computing & Human-Centered Technology, TU Wien, Austria. She received her PhD from The University of Tokyo, Japan in 2013. Her research interests include the algorithm development of customized graph representations, and she has been working on map labeling, railway map design, and complex network visualization.

Martin Nöllenburg is an associate professor for graph and geometric algorithms in the Algorithms and Complexity Group, TU Wien, Vienna, Austria. He received his PhD and habilitation degrees in computer science from Karlsruhe Institute of Technology (KIT) in 2009 and 2015. His research interests include graph drawing algorithms, computational geometry, and information visualization.