Real-time External Labeling of Ghosted Views

Ladislav Čmolík and Jiří Bittner

Abstract—We present a new algorithm for calculating the external labeling of ghosted views of moderately complex 3D models. The algorithm uses multiple criteria decision making, based on fuzzy logic, to optimize positions of the labels associated with different parts of the input model. The proposed method can be used with various existing algorithms for creating ghosted views from 3D models. The method operates in real-time, which allows the user to acquire a good understanding of the structure of the input model by studying the model and its labels from different viewpoints. We have conducted a user study to evaluate label layouts produced by our algorithm and those created by humans. The results show that the proposed method can significantly improve user understanding of labeled ghosted views of complicated 3D models, and its label layouts are comparable with label layouts created by humans.

Index Terms—External labeling, ghosted views, illustrative visualization, empirical evaluation, visualization for the masses.

1 Introduction

Complex objects composed of many distinct parts arise in various domains including mechanical engineering, biology, and medicine. Illustrations of such objects are essential in the process of communicating the spatial arrangement of the parts of the object. Numerous illustration methods are commonly used which include techniques such as *cutaways*, *exploded views*, *ghosted views*, and their combinations. Each of these methods provides a different way of understanding the structure of the model, and in turn, it is suitable for illustrating different types of models.

Ghosted views use transparency to depict the otherwise hidden parts of objects. The use of transparency is beneficial especially in situations in which we want to reveal several mutually occluding objects (see Figure 1). To create a ghosted view, the illustrator typically divides the parts of the illustrated object into semantically salient groups, paints each group into separate layers, specifies the layer transparencies and finally composes the layers into one image. Many techniques automate this process [9], [24] or use transparency to achieve similar effects to allow rendering of ghosted views from 3D models [12], [30] or volumetric data [8], [11].

The ghosted views alone cannot describe high-level semantic meaning and relationships between the model parts. The high-level relationships are thus provided verbally in the text which accompanies the illustration. The interconnection between the visual and the verbal information is mediated through labeling where labels, short textual annotations (e.g., names of the parts), are linked with the depiction of the parts. When dealing with ghosted views, the labeling problem becomes more complicated. In particular, the visual overlaps in the model make unambiguous linking of the labels to the model parts difficult. Thus we have to give the observer enough cues to understand which of the overlapping objects the given label belongs to. The existing automatic labeling methods do not explicitly consider transparency when computing the positions of labels and their anchors. This potentially leads to ambiguous results in which users can associate a given label with a wrong part of the model. In this paper we tackle this problem and aim at four main contributions:

1) We describe the first method for *external labeling of ghosted views*. The method takes the visual overlaps of the objects and their opacity into account during the process of optimizing positions of labels and their anchors.
2) We present *four new criteria* to determine salient parts of a semi-transparent 3D object. Labels are linked with the salient parts to make the labeling unambiguous.
3) We present a *user study* designed to evaluate how a label layout mediates the interconnection between the visual and verbal information for semi-transparent 3D objects. We compare label layouts produced by the presented algorithm with those produced by the state-of-the-art labeling algorithm for opaque objects [10] and those created by humans. The results show that the presented algorithm produces label layouts significantly better than the reference algorithm. The results of the proposed algorithm are comparable with the labelings created by humans.
4) The method has been implemented on the GPU and achieves real-time performance for moderately complex input models. The method rapidly calculates the labeling for the current frame achieving instantaneous reaction according to the classification of Nielsen [29, Section 5.5]. The movement of the labels is not temporally coherent between the frames. Therefore, we hide the labeling during interaction with the model.

2 Related Work

In external labeling the *label boxes* are placed outside of the area of illustration (denoted as *internal area*), and they are linked with the depiction of the parts using *leader lines*. The starting point of the leader line placed inside of the depiction of the labeled part is denoted as *anchor* (see Figure 1).

The task of the external labeling problem is to determine the *label layout*. That is, to determine such positions of the anchors and label boxes that exhibit *readability*, *unambiguity*, *aesthetics*, and *compactness* [2].

- *Ladislav Čmolík is with Faculty of Electrical Engineering at CTU in Prague. E-mail: cmolikl@fel.cvut.cz.*
- *Jiří Bittner is with Faculty of Electrical Engineering at CTU in Prague. E-mail: bittner@fel.cvut.cz.*

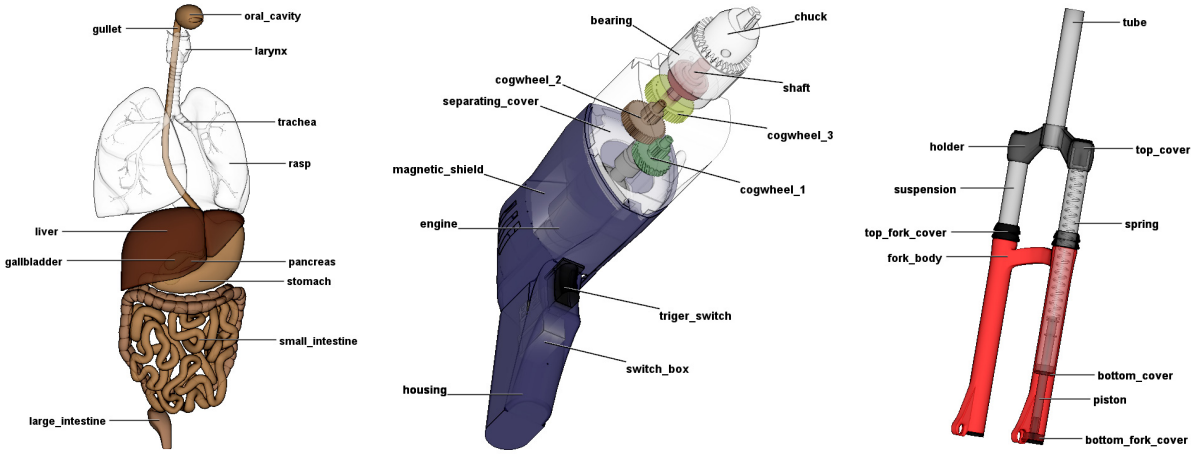


Fig. 1. Labeled ghosted views of digestive system, power drill, and front wheel fork of a bike using various label layouts. Our method performs optimization taking into account the opacity of depicted objects and thus it is able to compute unambiguous labeling of ghosted views.

We split the discussion of external labeling techniques into four parts according to the flexibility of anchors and label boxes (fixed vs. floating). In the case that the anchors or label boxes are fixed, their position is given as an input to the algorithm. In the case that they are floating, their positions have to be determined by the algorithm. Note, that only the labeling methods with floating anchors and floating label boxes can be considered as automatic as they do not require the positions of anchors, label boxes, or both as an additional input.

2.1 Fixed Anchors and Fixed Label Boxes

Bekos et al. [4] focused on the boundary labeling problem where fixed label boxes are arranged on the border of the internal area of rectangular shape enclosing a set of anchors. They study various types of leader lines, arrangements of label boxes and sizes of label boxes. Their primary focus is on efficient labeling algorithms that calculate leader lines whose combined length is minimal. Later, Benkert et al. [6] formulated the boundary labeling problem as a multiple criteria optimization problem where the length of leader lines, the number of bends, and the distance of anchors to leader lines are used to find an optimal solution of one-sided labeling where all label boxes are on one side of the enclosing rectangle.

These methods need positions of anchors and positions of label boxes as the input. The input is provided manually which can be a tedious process. Therefore, these methods are used for static illustrations only.

2.2 Floating Anchors and Fixed Label Boxes

Bekos et al. [3] extended the boundary labeling problem. The label boxes are again arranged on the border of the internal area of a rectangular shape, but each anchor can float within a polygonal area that is enclosed by the rectangular internal area. They propose efficient labeling algorithms for various types of leader lines under restrictions on the shape of the polygonal area and with the aim of minimizing the combined length of leader lines. The method is much more flexible in terms of anchor positions. However, the polygonal areas and the positions of label boxes are again provided manually. Thus, the method is again limited to labeling of static illustrations only.

2.3 Fixed Anchors and Floating Label Boxes

Preim et al. [32] addressed the two-sided boundary labeling problem where the label boxes can float on the left or right side of the internal area of a rectangular shape. They solve the problem for straight leader lines and also focus on temporal coherence of the label boxes when the anchors change their positions. Huang et al. [23] extended the approach of Bekos et al. [4] to support floating label boxes on the border of the internal area of rectangular shape for various types of leader lines. Stein and Décoret [36] presented a greedy algorithm for the labeling of fixed anchors attached to 3D objects. The occlusion of the 3D objects is minimized by placing label boxes in empty areas. Shadow regions and a summed area table [22] are used to prevent the crossing of leader lines and the overlapping of label boxes.

These methods can be utilized for labeling static and interactive illustrations. By interactive illustrations, we mean labeled renderings of a 3D model while a user is interacting with the model. In case of interactive illustrations anchors need to be manually attached on the surfaces of 3D objects. A limitation of these methods is that a 3D object will not be labeled if the attached anchor is occluded by another geometry even if part of the 3D object is visible in the rendered image.

2.4 Floating Anchors and Floating Label Boxes

Hartmann et al. [20] introduced a method to determine the labeling of 2D and 3D objects based on dynamic potential fields. The problem is split into first finding the anchors and then labeling using these anchors. The labeling is obtained as an equilibrium between attractive and repulsive forces established for the label boxes and the objects. Ali et al. [2] presented a real-time labeling pipeline, allowing the users to produce various labeling styles of 3D objects. The problem is again split into finding the anchors of 3D objects and labeling those anchors. This method is able to calculate the labeling of 3D models with frame-to-frame coherence at interactive frame rates. Čmolík and Bittner [10] presented a real-time algorithm, allowing the production of various labeling styles. Instead of splitting the problem into finding anchors and calculating the labeling for those anchors they directly calculate the leader lines that interconnect anchors and labels. The algorithm is able to compute the labeling of 3D models with frame-to-frame coherence at interactive frame rates. Götzelmann et al. [17]

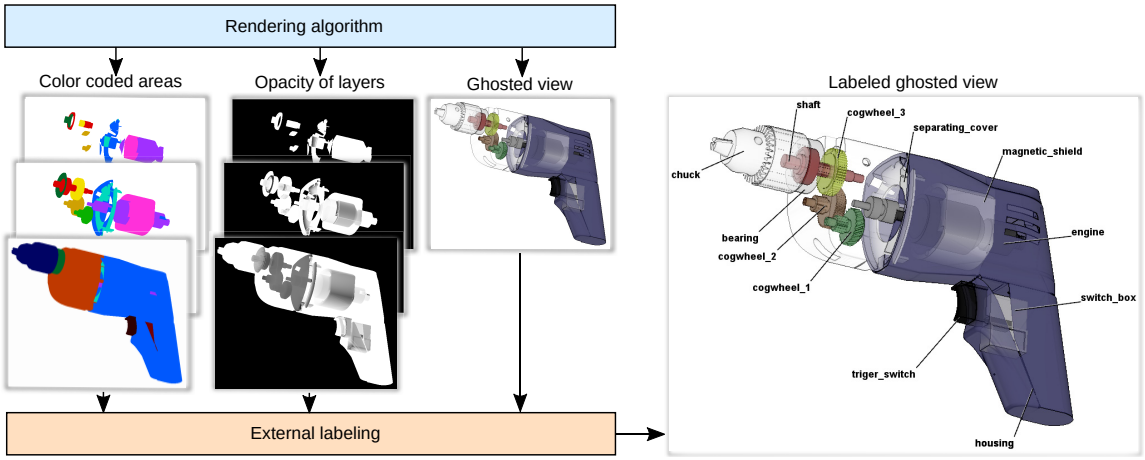


Fig. 2. An overview of our algorithm. The algorithm takes areas distributed over several layers with assigned opacity and an image of the ghosted view as an input and calculates label layout which is laid over the image of the ghosted view.

presented an agent-based labeling system, allowing the integration of internal and external labels. Also here the problem is split into finding anchors for 3D objects and labeling those anchors. Agents are assigned to initial labels and they compete and/or cooperate to meet metrics for functional and aesthetic label layouts extracted from handmade illustrations [21].

These methods are considered as automatic as they do not require the positions of anchors and label boxes as an additional input. The methods are primarily utilized to label interactive illustrations. They produce better results than methods with fixed anchors and floating label boxes as all visible geometry is labeled.

2.5 Other Methods

There are also methods that do not fit into the classification according to anchor and label box properties. The methods using external labeling for data points in a circular lens [7], [14], [15]. Luboschik et al. [27] use external labeling for point labeling in dense areas where all labels cannot be placed close to the corresponding point. The method of Götzelmann et al. [18] creates a contextual grouping of the labels. Götzelmann et al. [19] proposed a method for labeling animated objects. The method of Vollick et al. [38] can learn a specific labeling style from given examples and then apply the style to new illustrations. Oeltze-Jafra and Preim published a survey on labeling techniques used in medical visualization [31].

To the best of our knowledge only the technique of Mühler and Preim [28] takes the opacity of the 3D objects into account. Their technique considers occlusion of the opaque 3D objects by other 3D objects but labels only the opaque objects. Semi-transparent 3D objects are not labeled.

Using the above discussed automatic labeling methods for ghosted views in uniformed way produces ambiguous results. Since a given point in the ghosted view generally corresponds to a number of semi-transparent layers these methods are not able to optimize label positions for such layers.

Our proposed algorithm is the first to address the problem of automatically computing the labeling that is appropriate for ghosted views. In our algorithm all visible objects will be labeled without any need for additional information, and our labeling considers both the actual occlusion and opacity of the 3D objects.

3 External Labeling of Ghosted Views

This section presents a new labeling algorithm that computes the external labeling for a semi-transparent 3D model in real-time. First, let us define the problem by extending the definition of the external labeling to semi-transparent 3D models. A summary of basic labeling criteria of the presented algorithm taken from the state-of-the-art methods follows. Next, we describe our new criteria for the evaluation of anchor salience designed to lower the ambiguity of label layouts of ghosted views. Finally, we present the labeling algorithm.

3.1 Problem Description

We focus on the external labeling problem with floating anchors and floating label boxes where both the positions of anchors and label boxes have to be determined by the algorithm.

When solving the external labeling problem for an opaque 3D model we, in fact, deal with labeling 2D areas obtained by rendering the visible portions of the 3D model on the screen. Let us assume that the 3D model consists of a set of n 3D objects $\mathbf{O} = \{O_1, \dots O_n\}$. After rendering each object onto the screen and solving visibility, we obtain the projected areas $\mathbf{A} = \{A_1, \dots A_m, m \leq n\}$ of the visible parts of the objects. In the case of opaque objects these areas do not overlap and can be stored in a single layer.

In the case of rendering semi-transparent objects, the areas typically overlap. The overlapping can be eliminated by using several layers $\boldsymbol{\lambda} = \{\lambda_1, \dots \lambda_l\}$ (obtained for example with depth peeling [13]) in which the areas do not overlap. Each area may be located in several layers. Further, each pixel of each layer has its opacity (see Figure 2).

In the external labeling, the labels do not overlap with the projected areas $\mathbf{A}$. In our approach, we are placing the labels outside of *internal area* A_I enclosing the projected areas $\mathbf{A}$. We use a projection of the convex hull of the 3D objects $\mathbf{O}$ extruded to include a small boundary around the model as the internal area A_I (see Figure 3(a)). This approach is conservative and ensures that labels are placed outside of the projected areas $\mathbf{A}$. When the projection of the convex hull is used as the internal area, the resulting label layout will very likely exhibit compactness compared to a rectangular area enclosing the object. Nevertheless,

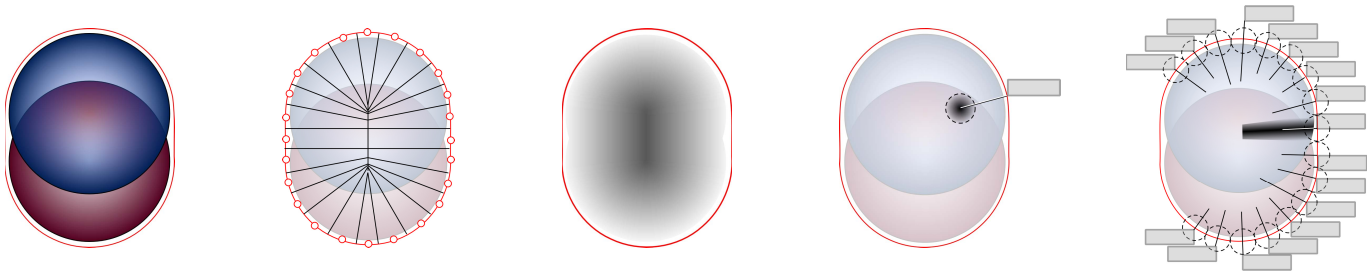


Fig. 3. (a) Two spherical 3D objects and the boundary of the internal area (red line) of the ghosted view calculated as extruded convex hull of the 3D objects. (b) Voronoi diagram calculated for several points on the boundary of the internal area. Each anchor candidate in the internal area is assigned one label box candidate when Voronoi diagram is calculated for each pixel of the boundary of the internal area. (c) Leader line length obtained from Voronoi diagram. Darker color means longer leader line. (d) When a position of anchor a_i is determined then anchor candidates within diameter d_1 from a_i are penalized based on the distance from a_i. (e) Positioning label boxes around the internal area. Label box candidates are points where leader lines intersect the boundary of internal area. Each label box touches the leader line with one corner. The corner of the label box is determined based on angle of the leader line from the x axis. When a position of Label box L_i is determined then anchor candidates whose associated label box candidates are within diameter d_2 from L_i are penalized based on the distance from L_i.

any convex shape enclosing the areas $\mathbf{A}$ can be used as the internal area A_I in our algorithm.

All pixels inside an area $A \in \mathbf{A}$ are considered candidates for the position of an anchor of a leader line pointing to the area A. Similarly, all pixels on the boundary of internal area A_I are considered candidates for the position of a label box at the end of the leader line pointing to the area A. The label box candidate is the pixel where the leader line intersects the boundary of the internal area, see Figure 3(e).

The task of the external labeling is to determine the *label layout*. That is, to determine the positions of the anchors and label boxes for all areas in $\mathbf{A}$ such that the labels are readable, each label is close to the labeled area, and each label is unambiguously associated with the labeled area.

The quality of label layout is evaluated according to a number of criteria for anchors, label boxes, and leader lines. We use four criteria that are used by the state-of-the-art techniques [2], [10], [21] and define four new criteria related to the semi-transparency.

We formulate the external labeling as a multiple criteria optimization problem where we search for the label layout. In other words, for each area in $\mathbf{A}$ we have to choose an anchor and label box from the available candidates that satisfy the criteria best. The following section summarizes the basic criteria used in the presented algorithm.

3.2 Basic Labeling Criteria

This section describes the labeling criteria of the presented algorithm taken from the state-of-the-art methods.

Leader line direction

To satisfy this criterion we utilize the approach of Čmolík and Bittner [10] where the leader line for each anchor candidate is determined by calculating the Voronoi diagram of the boundary of the internal area A_I, see Figure 3(b). This way, one label box candidate $\vec{L}$ on the boundary of A_I is assigned to each anchor candidate and the leader line is determined for each anchor candidate as the line between the anchor candidate and its assigned label box candidate. By using different distance metrics in the calculation of the Voronoi diagram, e.g., limiting the directions for which the nearest point is evaluated, the approach is able to limit also the directions of leader lines which allows us to create various label layout styles, see Figure 1.

The approach evaluates the length of each leader line in the process of calculating the Voronoi diagram, see Figure 3(c). Further, the approach ensures that the closest label box candidate on the boundary of the internal area A_I, and thus the shortest possible leader line, is assigned to each anchor candidate.

Leader line length

Leader lines should be short as possible. This ensures that the labels will be close to the labeled areas. We use the approach of Čmolík and Bittner [10] to model how each anchor candidate complies with this criterion as

$$f_1(\vec{a}) = 1 \quad \frac{dist(\vec{a},\vec{L})}{d_{max}} \tag{1}$$

where $dist(\vec{a},\vec{L})$ is the distance between anchor candidate $\vec{a}$ and its associated label box candidate $\vec{L}$ (e.g., length of the leader line) and d_{max} is the longest distance between an anchor candidate and its associated label box candidate (e.g., the longest leader line).

Anchor distance

Anchors should not be close together, otherwise it could be hard to determine to which area the leader line is pointing. To model this criterion we utilize the approach of Ali et al. [2]. Anchor candidate $\vec{a}$ within diameter d_1 to the already determined anchors $\mathbf{a}_d$ is penalized based on the equation

$$f_2(\vec{a}) = \prod_{\vec{a}_d \in \mathbf{a}_d} f_2^{'}(\vec{a},\vec{a}_d) \tag{2}$$

$$f_2^{'}(\vec{a},\vec{a}_d) = \begin{cases} dist(\vec{a},\vec{a}_d)/d_1; & dist(\vec{a},\vec{a}_d) < d_1 \\ 1; & dist(\vec{a},\vec{a}_d) \geq d_1 \end{cases} \tag{3}$$

where $dist(\vec{a},\vec{a}_d)$ is the distance between anchor candidate $\vec{a}$ and the already determined anchor $\vec{a}_d$, see Figure 3(d). We use $d_1 = 0.18$ a value recommended by Čmolík and Bittner [10].

Label box distance

Label boxes should not overlap, otherwise they would not be readable. We use the approach of Čmolík and Bittner [10] to model this criterion. Anchor candidate $\vec{a}$ whose associated label

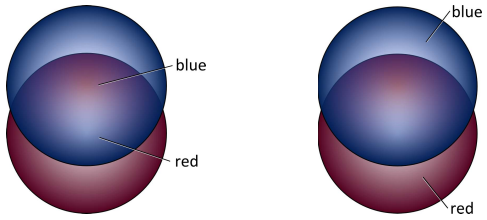


Fig. 4. (a) Ambiguous labeling of two semi-transparent objects. (b) Unambiguous labeling of the same objects.

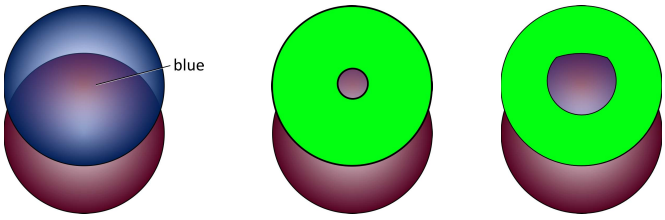


Fig. 5. (a) Anchor of "blue" label is in the overly transparent area of the blue object. The labeling is ambiguous. (b and c) Area (green) of the blue object where the opacity is greater or equal to 0.15 (b) and 0.25 (c).

box candidate is within diameter d_2 to the label boxes of the already determined anchors $\mathbf{a}_d$ is penalized based on the equation

$$f_3(\vec{a}) = \prod_{\vec{a}_d \in \mathbf{a}_d} f_3^{'}(\vec{a}, \vec{a}_d) \tag{4}$$

$$f_3^{'}(\vec{a}, \vec{a}_d) = \begin{cases} dist(\vec{L}, \vec{L}_d)/d_2; & dist(\vec{L}, \vec{L}_d) < d_2 \\ 1; & dist(\vec{L}, \vec{L}_d) \geq d_2 \end{cases} \tag{5}$$

where $\vec{L}$ is label box candidate associated with anchor candidate $\vec{a}$, $\vec{L}_d$ is label box of already determined anchor $\vec{a}_d$, $dist(\vec{L}, \vec{L}_d)$ is the distance between label box candidate $\vec{L}$ and the label box $\vec{L}_d$, see Figure 3(e). Ali et al. [2] shows that if the diameter d_2 is large enough then the label boxes can be positioned around a convex internal area without overlapping except for the top and bottom parts of the internal area where we often have to prolong the leader line to prevent the overlapping, see Figure 3(e). We use $d_2 = 0.05$, a value recommended by Čmolík and Bittner [10].

Leader line crossing

Leader lines should not cross, otherwise it could be hard to follow where the individual leader lines are pointing. This holds especially in cases where the crossing leader lines have a similar direction. We use the approach of Čmolík and Bittner [10] to satisfy this criterion. If the *leader line direction* and *label box distance* criteria are satisfied then the leader lines cannot cross or overlap.

3.3 Anchor Salience Criteria

When solving the external labeling problem we want the anchors to be salient points of their areas. In other words, we want the leader line to point to such a part of the area so that the user will clearly associate the corresponding label with the correct 3D object. If the anchor is not a salient point of the area, then the label layout may be ambiguous and the user is unlikely to associate the label with the correct 3D object. In the state-of-the-art algorithms for external labeling of opaque 3D objects [2], [10], the salience of an anchor candidate of an area was defined as the shortest distance between the anchor candidate and the outline of the area. That is because the areas do not overlap and placing the anchor close to the boundary of the area can lead to ambiguous label layout where the user associates a wrong area (e.g., the one on the other side of the boundary) with the label.

When dealing with the external labeling of ghosted views the situation is more complicated since the 3D objects, and consequently, the areas may overlap one another or themselves. The layers have associated opacity, and the anchors on the same position, but in different layers should not be treated independently. An example of ambiguous label layout that only optimizes the distance of the anchors from the boundary is shown in Figure 4(a).

Thus, we need to define what is a salient point of an area to obtain unambiguous label layouts such as the one in Figure 4(b). Similarly as with opaque 3D objects, in ghosted views we want the leader line to point to such a part of the semi-transparent 3D object so that the user will clearly associate the label with the semi-transparent 3D object. To do so, we introduce four new criteria for the anchors.

Opacity salience

Anchors should not be placed in an overly transparent part of an area on the layer λ_i. Otherwise the user may associate the label with another area (3D object) on a lower layer $\lambda_j, j > i$ that is visible through the layer λ_i (see label *blue* in Figure 5(a)).

We define the opacity salience of an anchor candidate $\vec{a}$ as:

$$f_{s_1}(\vec{a}) = \begin{cases} 0; & \alpha_{\vec{a}} < T_{s_1} \\ 1; & \alpha_{\vec{a}} \geq T_{s_1} \end{cases} \tag{6}$$

where T_{s_1} is the threshold for the opacity of the anchor and $\alpha_{\vec{a}}$ is the opacity at the layer and position at which the the candidate is located. With this criterion we disqualify anchor candidates whose opacity is lower than the threshold T_{s_1}. With a greater value of the threshold T_{s_1} we disqualify more anchor candidates, see Figures 5(b) and 5(c). In our implementation we use $T_{s_1} = 0.25$. We have determined the value of the threshold by experimenting with its value and observing the resulting label layouts. If a leader line is pointing to the area with opacity lower than 0.25 then the user is unlikely to associate the label with the correct 3D object.

Occlusion salience

Anchors should not be placed in such parts of an area on the layer λ_i where the accumulated opacity of the upper layers $\lambda_j, j < i$ is

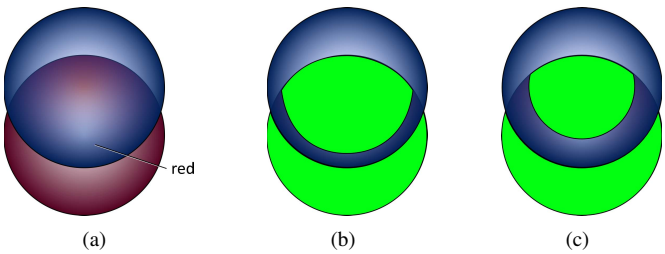


Fig. 6. (a) Anchor of "red" label is in the overly occluded area of the red object. The labeling is ambiguous. (b and c) Area (green) of the red object where the occlusion is lower or equal to 0.95 (b) and 0.9 (c).

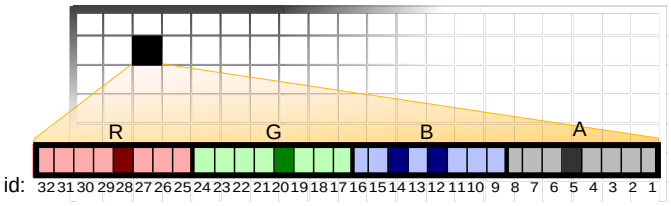


Fig. 7. The 2D binary mask allows the storage of multiple ids of areas as bits on unique positions in RGBA pixel. Here illustrated with 8 bits per channel, however in our implementation we use 32 bits per channel. The highlighted pixel stores the following ids: 5, 12, 14, 20, and 28.

high. Otherwise, the label may be associated with another area on one of the upper layers $\lambda_j, j < i$ as the layer λ_i is not clearly visible through them (see label *red* in Figure 6(a)).

We define the occlusion salience of an anchor candidate $\vec{a}$ located on layer λ_j as:

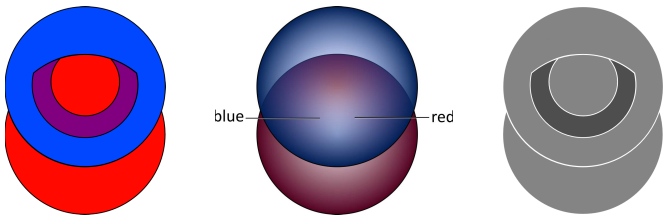
$$f_{s_2}(\vec{a}) = \begin{cases} 0; & \Omega_{\vec{a}} > T_{s_2} \\ 1; & \Omega_{\vec{a}} \leq T_{s_2} \end{cases} \tag{7}$$

where T_{s_2} is the threshold for the occlusion opacity and $\Omega_{\vec{a}} = 1 \prod_{k=1}^{j} 1 \quad \alpha_k$ is the accumulated opacity of the layers $\lambda_k, k \in 1 \dots j$, α_k is the opacity of layer λ_k at the position of the anchor candidate. With this criterion we disqualify anchor candidates whose occlusion is greater than the threshold T_{s_2}. With a lower value of the threshold T_{s_2} we disqualify more anchor candidates, see Figures 6(b) and 6(c). In our implementation we use $T_{s_2} = 0.9$. Again, we have determined the value of the threshold by experimenting with its value and observing the resulting label layouts. If a leader line is pointing to an area with an accumulated opacity higher than 0.9 then the user is unlikely to associate the label with the correct 3D object.

Encoding opacity salience and occlusion salience

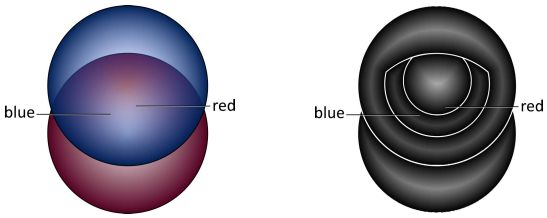
First, let us describe how we encode the *opacity salience* and *occlusion salience* criteria into a binary mask texture (bitmask). Later, we use the bitmask, to define *overlap salience* and *outline salience*, the remaining two anchor salience criteria.

The anchor candidate $\vec{a}$ of area A_i is suitable for the area only if $\tilde{f}_{s_1}(\vec{a}) \wedge_B \tilde{f}_{s_2}(\vec{a}) = 1$ where $\wedge_B$ is a Boolean conjunction. We store this information in the bitmask (see Figure 7). Each unique id of an area is stored in the bitmask as one bit on a unique position in an RGBA pixel. We use 32bit unsigned integer RGBA textures



(a) (b) (c)

Fig. 8. (a) Visualization of the bitmask. Blue color indicates the area where the blue object is clearly visible, red color indicates the area where the red object is clearly visible, purple color indicates the area where both objects are clearly visible. (b) Anchors of both "blue" and "red" labels are in the purple area. The labeling is ambiguous. (c) Visualization of the overlapping areas. Dark color indicates several overlapping areas.



(a) (b)

Fig. 9. (a) Anchors of the labels are close to the discontinuity in the bitmask. One of the anchors is in the area where more than one objects are clearly visible. The labeling is ambiguous. (b) Visualization of the shortest distance form the outlines detected as discontinuities in the bitmask. Darker color indicates shorter distance.

which allow us to store up to 128 (4×32) unique ids in one pixel of the texture. This number was satisfactory for our experiments. Texture arrays can be used if more unique ids need to be stored. We use the bitmask to mask those anchor candidates which are not suitable for a selected area.

Note that both the opacity of the layer and the accumulated opacity of previous layers are available in algorithms for rendering semi-transparent 3D models in front-to-back order as depth peeling [13] or concurrent linked lists [39]. Therefore, we evaluate these criteria already, during the rendering process. If we obtained only the layers and their opacity as the input, then we would have to calculate the *opacity salience* and *occlusion salience* criteria by traversing the layers in front-to-back order.

Overlap salience

Anchors should not be placed where many areas in the bitmask overlap. The fewer areas are overlapping at the position of the anchor the higher is the chance that the user will associate the label with the correct 3D object.

This criterion penalizes parts of the layers where many areas overlap and thus moves the anchor away of such parts. As we select the thresholds T_{s_1} and T_{s_2} for opacity salience and occlusion salience for all areas, and not individually for each area, it may not be always possible to select thresholds suitable for all areas. This criterion helps improve the label layout in such cases. Figure 8(a) shows a visualization of the bitmask where anchors for both red and blue objects can be selected from anchor candidates in the purple area. If this happens we end up with ambiguous labeling on Figure 8(b).

We define the overlap salience of an anchor candidate $\vec{a}$ as:

$$f_{s_3}(\vec{a}) = \frac{k}{m} \tag{8}$$

where k is the number of areas stored in the bitmask at the position of anchor candidate $\vec{a}$ and m is the number of the layers. See Figure 8(c) for an example.

Outline salience

The anchors should not be placed close to outlines detected as discontinuities in the bitmask storing the clearly visible areas.

An outline detected as the discontinuity in the bitmask indicates that there are different areas on each side of the outline and when an anchor is placed close to such an outline the user is unlikely to associate the label with the correct 3D object. This is mainly a problem if more than one area is visible at least at one

side of the outline (see Figure 9(a)). We define the outline salience of an anchor candidate $\vec{a}$ as

$$f_{s_4}(\vec{a}) = \frac{dist_{out}(\vec{a})}{d_{max}} \tag{9}$$

where $dist(\vec{a})$ is the shortest distance from $\vec{a}$ to the detected outline. The distance is scaled into the range $[0,1]$ by dividing it by the length of the longest leader line d_{max} which represents the maximal possible distance between two samples. Note that it is possible to calculate the distance from all anchors to the closest outline in one step with the jump flooding algorithm [33]. See Figure 9(b) for an example.

Note that all the criteria can be evaluated for all anchor candidates in the internal area A_I in one step and the obtained values can be stored in 2D textures. The values returned by evaluating the criteria are the same for all anchor candidates at the same position even though they are on different layers. Thus we can replace all such anchor candidates with one anchor candidate and use the bitmask to mask the anchor candidates that are outside the desired area A_i.

3.4 Calculating Label Layout

We use greedy optimization to determine the label layout and a heuristic to determine the order in which areas are labeled. The optimization proceeds as follows:

1) Determine leader lines for all anchor candidates in A_I.
2) Evaluate criteria for each anchor candidate in A_I.
3) Aggregate criteria for each anchor candidate in A_I.
4) While there is an unlabeled area do:
   a) Select an unlabeled area A_s.
   b) Select the best anchor candidate a_s for the area A_s.
   c) Update each anchor candidate.
5) If needed, correct the label layout.

In the following text, we describe the individual steps of the algorithm in more detail.

Determining Leader Lines for All Anchor Candidates

To determine the leader line for each anchor candidate we utilize the approach of Čmolík and Bittner [10] where one label box candidate, i.e., the nearest point on the boundary of the internal area A_I, is associated to each anchor candidate. The label box candidate and the leader line of each anchor candidate are determined by calculating the Voronoi diagram of the boundary of the internal area A_I. By limiting the directions for which the nearest point is evaluated, the approach limits also the directions of the leader lines. Therefore, we satisfy the *leader line direction* criterion in the process of calculating the Voronoi diagram. Further, the *leader line length* criterion of the leader lines is determined for each anchor candidate. We store the associated label box candidates and the length of the determined leader lines in a 2D texture.

Evaluating Criteria for Anchor Candidates

For each anchor candidate, we use the precomputed values of the *leader line length* criterion from the 2D texture created in the previous step and the precomputed values of the *opacity salience* and *occlusion salience* criteria from the bitmask. If we obtained the layers and their opacity as the input instead of the bitmask, then we would need to encode the *opacity salience* and *occlusion salience* criteria into the bitmask by traversing the layers in front-to-back order first.

We evaluate the *overlap salience* by counting the bits in the bitmask for each anchor candidate and store it in a 2D texture. Further, we detect outlines as discontinuities in the bitmask and evaluate the *outline salience* criterion with the jump flooding algorithm [33] and again store it in a 2D texture.

Aggregating Criteria for Anchor Candidates

Our method uses multiple criteria, which are used to evaluate the anchor candidates. We search for simultaneous satisfaction of these, possibly contradicting, criteria. In this process, we use fuzzy set theory, fuzzy logic, and fuzzy optimization [40] based on fuzzy decision making by Bellman and Zadeh [5].

Fuzzy set theory is an extension of the set theory. Unlike the set theory, the fuzzy set theory can express a partial membership of an element in the set. A fuzzy set is commonly described by its membership function that maps each element to values in the range $[0,1]$ which indicates the membership of the element in the set, 0 means that the element is not in the set and 1 means that the element is entirely in the set. Fuzzy logic defines operations on the fuzzy sets that are equivalents of Boolean logic operations (e.g., negation, conjunction, and disjunction).

In the fuzzy optimization, we consider the solution space X containing all anchor candidates in internal area A_I. Each criterion C_k for evaluation of anchor candidates is modeled as a fuzzy set on A_I, and membership function f_k describes the satisfaction of the criterion C_k by the anchor candidates $\vec{a}_i \in A_I$. To obtain simultaneous satisfaction for q criteria $C_k, k = 1 \dots q$ the membership functions f_k of individual criteria are aggregated together using a fuzzy conjunction:

$$f(\vec{a}_i) = \bigwedge_{1 \le k \le q} f_k(\vec{a}_i)^{w_k} \tag{10}$$

Weights $w_k, k = 1 \dots q$ are used to balance the influence of individual criteria. Note that the fuzzy conjunction guaranties the simultaneous satisfaction of the criteria. In other words, satisfaction of one criterion cannot compensate dissatisfaction of another criterion. We use natural T-norm as the fuzzy conjunction, which corresponds to standard multiplication.

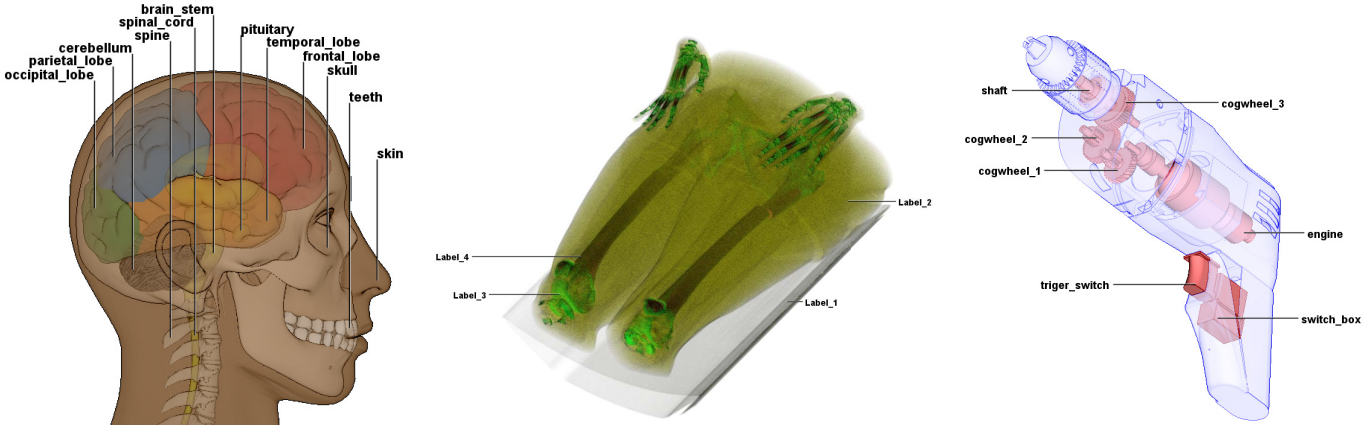
In this step, we aggregate the *leader line length*, *overlap salience*, and *outline salience* criteria for evaluation of anchor candidates in Equation 10 and store the result in 2D texture.

Selecting Unlabeled Area

We use the aggregated criteria and the bitmask, described in Section 3.3, to select one of the areas for labeling. For each unlabeled area, we mask anchor candidates that are not suitable using the bitmask (in other words, we evaluate the anchor candidates according to the *opacity salience* and *occlusion salience* criteria) and calculate the sum of the aggregated criteria for all suitable anchor candidates. We label the area for which the sum is the lowest first as placing an anchor, a leader line, and a label box over the ghosted view reduces the number of suitable anchor candidates. We denote the selected area as A_s.

Selecting Best Anchor Candidate

For the selected area A_s we mask the anchor candidates that are not suitable using the bitmask (we again evaluate the anchor candidates according to the *opacity salience* and *occlusion salience* criteria). From the remaining suitable anchor candidates,



(a) (b) (c)

Fig. 10. (a) Ghosted view of human head with one sided label layout where the leader lines follow the vertical direction. (b) Ghosted view produced with Direct Volume Rendering. The color-coded areas of the volume are labeled with the proposed algorithm. (c) Ghosted view produced with the approach of Nienhaus and Döllner [30] labeled with the proposed algorithm.

we select the anchor candidate for which the aggregate criteria yields maximum value and use it as anchor $\vec{a}_s$ of the area A_s. The label box candidate associated with the anchor is used as label box $\vec{L}_s$ of the area A_s. We mark the area A_s as labeled.

Updating Anchor Candidates

In this step we update the texture storing the aggregated criteria of anchor candidates computed using Equation 10. We decrease the value stored in the texture by multiplying the value with result of Equation 3 where we use the anchor $\vec{a}_s$ selected in the previous step as the already determined anchor $\vec{a}_d$. This decreases the value of those anchor candidates that are in the diameter d_1 around the anchor a_s, see Figure 3(d). At this point all anchor candidates are also evaluated according to the criterion *anchor distance*. Similarly, we decrease the value stored in the texture by multiplying the value with result of Equation 5 where we again use the anchor $\vec{a}_s$ selected in the previous step as the already determined anchor $\vec{a}_d$. This decreases the value in the texture of those anchor candidates that have an associated label box candidate positioned in the diameter d_2 around label box $\vec{L}_s$ selected in the previous step, see Figure 3(e). At this point all anchor candidates are also evaluated according to the criterion *Label box distance* and we also satisfy the *leader line crossing* criterion.

Correcting the Label Layout

If diameter d_2 from the previous step is too small then the label boxes may overlap. If this is the case we reposition them with the approach of Ali et al. [2].

4 Results and Discussion

We evaluated our method using a test application implemented in Java and OpenGL. For measurements, we used a PC with Intel Xeon CPU E5-1620 at 3.6GHz, 16GB of RAM, and NVIDIA GeForce GTX 470 with 4GB of RAM. We used four test models consisting of 75k to 215k triangles with 1 to 35 object parts associated with labels. The performance of the labeling algorithm is shown in Figure 11. For all tested models the implementation delivers real-time performance – please see the accompanying video for a live capture of our application.

For all label layouts in the paper and also in the user study described below we have used the same parameters. The diameters d_1 and d_2 are specified in the texture space (assuming the texture is a unit square); we used $d_1 = 0.18$ and $d_2 = 0.05$. The thresholds, for opacity and accumulated opacity are $T_{s_1} = 0.25$ and $T_{s_2} = 0.9$. The weights used for the criteria aggregated in Equation 10 are $w_L = 1.11$ for the *leader line length* criterion, $w_{s_3} = 5$ for the *overlap salience* criterion, and $w_{s_4} = 1.95$ for the *outline salience* criterion.

We have used an extended approach of Krüger et al. [24] to generate the ghosted views. Nevertheless, the presented algorithm can be used with any algorithm capable of producing the areas distributed over several semi-transparent layers. Our approach is able to compute label layouts in which leader lines follow different directions (see Figures 1 and 10).

We have tested our method with ghosted views produced by the method of Krüger et al. [24], direct volume rendering of Levoy [25] (see Figure 10(b)) and the method of Nienhaus and Döllner [30] (see Figure 10(c)).

In Figure 10(c) we also demonstrate that the algorithm is capable of labeling a subset of selected 3D objects. Note that the rendering algorithm needs to produce areas for all 3D objects, even for those that will not be labeled. The reason is that the unlabeled areas may influence (e.g., occlude) the areas that will be labeled and the labeling algorithm needs this information. To label only the selected 3D objects, our labeling algorithm will obtain the set of ids of areas to label as an additional input and use this set to select the area for labeling in step 4a of our algorithm.

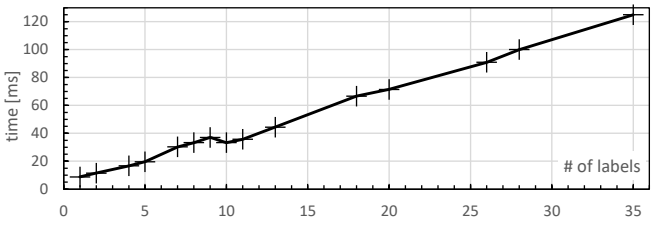


Fig. 11. Time needed for calculation of the labeling depending on the number of labels using 5 different 3D models. The labeling was calculated in an off-screen buffer (512x512).

It is also possible to label several 3D objects with only one label, simply by assigning the same id to all objects that should be labeled together. The id will be propagated to the corresponding areas. Figure 1 (right) shows an example where the same parts on the left and right of the front bike wheel fork are labeled only once.

Our approach has several limitations. When the leader lines follow the vertical direction (as in Figure 10(a)), they have to be prolonged to prevent overlaps of the label boxes. In such cases the layout of the label boxes could be improved with the approach of Gemsa et al. [16]. Further, a color of a 3D object in a semi-transparent 3D model could be changed in the blending process which could result in a ghosted view where several objects have similar color, and it is harder to visually determine boundaries of the objects and the correspondence of labels and 3D objects. Lastly, our approach does not support temporal coherence of the anchors and label boxes. We have tested additional criteria for temporal coherence from previous work [2], [10], but due to the many discontinuities in the bitmask storing the clearly visible parts of the areas the movement of anchors and label boxes was not coherent. Therefore, we do not show the label layout when a user is rotating the 3D model. We believe that temporal coherence of the anchors and the label boxes can be improved during the rotation of the 3D model with the approach of Tatzgern et al. [37].

5 Empirical Evaluation

The purpose of labeling is to mediate an unambiguous interconnection between visual and verbal information. In the case that the labeling is ambiguous, the viewer may associate verbal information (e.g., name of the object) with the wrong depiction of the object. We have conducted an evaluation with users to assess the influence of the introduced salience criteria on the ambiguity of the calculated labeling. We recruited 60 participants (13 females), all daily users of computers, with age ranging from 20 to 61 years (mean = 25.1; SD = 7.52).

For the evaluation, we created a web application which the participants accessed through a web browser. First, each participant was instructed about the testing procedure; then the participant provided her/his age and gender. An image with a labeled 3D model was presented to the participant, and the task of the participant was: (1) Examine the image and for each label try to determine to which part of the depicted object it belongs. (2) Press *Start test* button and wait. One of the object parts was highlighted in light green. (3) Click on the label that belongs to the highlighted object part as quickly as possible, but at the same time try to be sure that the label belongs to the highlighted object part. If the participant could not decide which label belongs to the highlighted object part then s/he pressed the *I cannot decide* button. If the participant thought that there is no label for the highlighted object part then s/he pressed the *There is no label* button. (4) Wait until another object part is highlighted, and continue as before.

The whole test was repeated on four different images of labeled 3D models. For each 3D model, we chose a representative view from which all labeled parts were clearly visible and used labeling calculated with our algorithm or created by a human. The sequence in which the objects in the 3D model were highlighted was predetermined and it was the same for all labeling methods. The complexity of the ghosted view is varying for the 3D models. For the *digestive* model the ghosted view contains only a few overlaps of the 3D objects. For the *drill* model the overlaps of the 3D objects are more complex, and for the *fork* and *head* model the 3D objects overlap heavily.

Two variants of our algorithm were used in the study that differ in the criteria that were used. The labeling method *M0* uses the *leader line length*, *label box distance*, *anchor distance*, and *outline salience* criteria. The labeling method *M1* corresponds to the proposed algorithm and uses the *leader line length*, *label box distance*, *anchor distance*, *outline salience*, *overlap salience*, *opacity salience*, and *occlusion salience* criteria. Note that the labeling method *M0* corresponds to the state-of-the-art method [10], but instead of the originally proposed salience criterion used in the method we use the outline salience criterion. We made this change to allow labeling of semi-transparent objects.

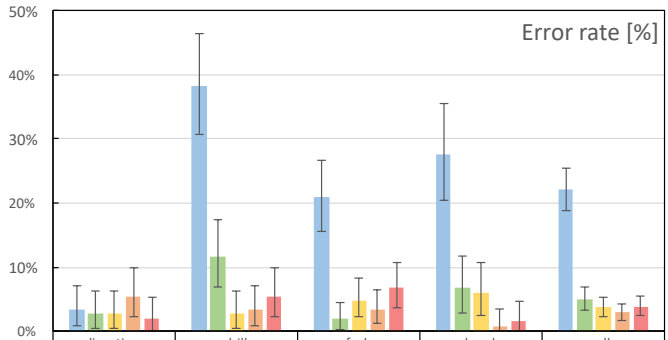
Further, label layouts created by three humans were used in the study to compare label layouts created by our algorithm with label layouts created by humans. Despite our efforts, we were not able to obtain label layouts from professional illustrators for the comparison. Therefore, we have recruited experts who work in related fields (user interface and graphics design). This fact should be taken into account into account when interpreting the results of the evaluation. The first (*P1*) and second (*P2*) label layouts were created by experts in user interface design. The third label layout (*P3*) was created by an expert in graphics design. All of the experts are long time workers in their respective fields.

We measured the *completion time*, measured as a sum of times between highlighting of an object part and clicking on a label, and the *number of errors* as the number of wrongly selected labels. When the participant could not decide which label belongs to the highlighted object part or thought that the highlighted object part is not labeled (it always was) we counted this as an error.

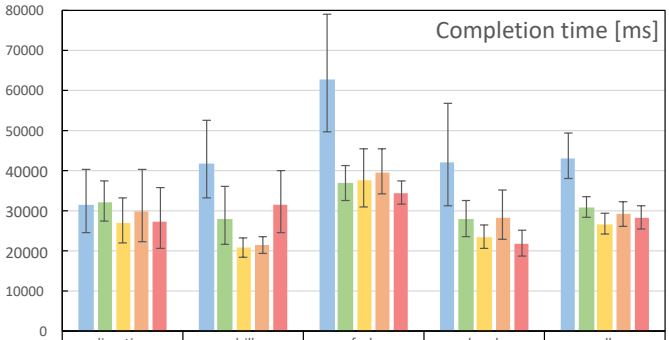
The experiment was one factor with five levels. The independent variable was the labeling method (*M0*, *M1*, *P1*, *P2*, and *P3*). One participant was tested only once for each of the four 3D models. For each of the 3D models, the participant was tested with a different labeling method. Both the 3D models and the labeling methods were counterbalanced. The sequences in which the 3D models were presented to the participants were determined as rows of 4×4 balanced Latin square. For each sequence of 3D models, five sequences of labeling methods were determined as rows of 5×5 Latin square in which only the first four columns were used. That gives us 20 sequences of 3D model and labeling method pairs where each pair is repeated exactly four times, each time on different position, and each pair precedes another pair exactly once. Each sequence of the pairs was repeated three times to obtain 60 sequences.

Participants were tested only once for each of the four 3D models to eliminate the learning effect and fatigue. There is a possibility that the participant remembers the position of a label from one test and intentionally or unintentionally uses it when s/he is tested with a different technique. Then, both the number of errors and the completion time would be influenced. This could be the case for the *head* model where some label layouts are quite similar, see the supplementary material of this paper. Further, if the participant is required to repeat the study on the same 3D model multiple times then there is a possibility of influencing the results by fatigue. The counterbalancing was used to eliminate ordering and carry-over effects.

We evaluated the collected data for each 3D model separately and for all 3D models together. We tested each participant with a different labeling method for each of the four 3D models to have independent groups of participants when we evaluate all 3D mod-



	digestive	drill	fork	head	all
M0	3.42 (0.85; 7.16)	38.36 (30.66; 46.34)	20.87 (15.58; 26.68)	27.61 (20.38; 35.45)	22.04 (18.88; 25.37)
M1	2.74 (0.44; 6.22)	11.64 (6.86; 17.38)	1.94 (0.30; 4.43)	6.72 (2.93; 11.68)	4.95 (3.37; 6.80)
P1	2.74 (0.44; 6.22)	2.74 (0.44; 6.22)	4.85 (2.22; 8.29)	5.97 (2.40; 10.73)	3.67 (2.31; 5.31)
P2	5.48 (2.20; 9.87)	3.42 (0.85; 7.16)	3.40 (1.21; 6.41)	0.75 (0.00; 3.40)	2.88 (1.68; 4.35)
P3	2.05 (0.06; 5.24)	5.48 (2.20; 9.87)	6.80 (3.67; 10.69)	1.49 (0.00; 4.59)	3.83 (2.44; 5.50)



	digestive	drill	fork	head	all
M0	31 576 (24 723; 40 329)	41 865 (33 406; 52 466)	62 703 (49 865; 78 846)	42 280 (31 434; 56 869)	43 267 (37 997; 49 269)
M1	32 051 (27 373; 37 528)	28 002 (21 712; 36 114)	36 854 (32 811; 41 396)	27 854 (23 767; 32 644)	30 981 (28 490; 33 691)
P1	27 145 (22 184; 33 216)	20 859 (18 566; 23 434)	37 621 (31 059; 45 569)	23 577 (20 904; 26 592)	26 621 (24 160; 29 332)
P2	29 955 (22 237; 40 351)	21 500 (19 495; 23 711)	39 507 (34 211; 45 624)	28 374 (22 880; 35 186)	29 149 (26 195; 32 437)
P3	27 373 (20 829; 35 974)	31 395 (24 556; 40 139)	34 468 (31 602; 37 593)	21 788 (18 805; 25 244)	28 344 (25 668; 31 299)

Fig. 12. A comparison of average error rates (left) and average completion times (right) for the labeling methods. The error bars show the 95% confidence intervals. If the confidence intervals do not overlap then the means of the measured data are significantly different.

els together. We performed a statistical evaluation of the measured data using confidence intervals. We transformed the measured number of errors into error rates with the LaPlace method recommended by Lewis and Sauro [26] and calculated the confidence intervals of the error rates as adjusted Wald intervals, a method recommended for completion rates [1], [34]. We calculated the confidence intervals of measured completion times as confidence intervals for task completion times [35, Chapter 3]. When the confidence intervals do not overlap, we can report that the mean values of the measured data are significantly different. We use 95% confidence intervals for both error rates and completion times. If we repeated the experiment many times with different samples from a population then 95% of the calculated confidence intervals would contain the true mean of the population.

The average error rates and completion times together with their 95% confidence intervals are shown in Figure 12. The labeled representative views for the *digestive*, *drill*, and *head* models used in the evaluation are shown in the supplementary material of this paper. The labeled representative views for the *fork* model used in the evaluation are shown in Figure 13. These figures also depict the average error rates of individual labels.

The comparison of error rates shows that for the digestive model there is no significant difference between the error rates of the *M0*, *M1*, *P1*, *P2*, and *P3* labeling methods. This can be expected as the model is simple and there are only a few overlapping objects in its ghosted view. For the drill model the label layouts produced with the *M1*, *P1*, *P2*, and *P3* labeling methods provide significantly lower error rates than the label layout produced with the *M0* method, the label layout produced with the *P1* method provides significantly lower error rate than the label layout produced with the *M1* method, there is no significant difference between the error rates of the *M1*, *P2*, and *P3* labeling methods. For the fork and head models the label layouts produced with the *M1*, *P1*, *P2*, and *P3* labeling methods provide significantly lower error rates than label layouts produced with the *M0* method, there is no significant difference between the error rates of the *M1*, *P1*, *P2*, and *P3* labeling methods.

The comparison of error rates for all models evaluated together shows that the label layouts produced with the *M1*, *P1*, *P2*, and *P3* labeling methods provide significantly lower error rates than label layouts produced with the *M0* method. There is no significant difference between the error rates of the *M1*, *P1*, *P2*, and *P3* labeling methods.

The comparison of completion times shows that for the digestive model there is no significant difference between the completion times of the *M0*, *M1*, *P1*, *P2*, and *P3* labeling methods. For the drill model the label layouts produced with the *P1* and *P2* labeling methods provide a significantly lower completion time than label layouts produced with the *M0* and *P3* labeling methods, there is no significant difference between the completion times of the *M1*, *P1* and *P2* labeling methods. For the fork model the label layouts produced with the *M1*, *P1*, *P2*, and *P3* labeling methods provide a significantly lower completion times than label layouts produced with the *M0* method, there is no significant difference between the completion times of the *M1*, *P1*, *P2*, and *P3* labeling methods. For the head model the label layouts produced with the *P1* and *P3* labeling methods provide a significantly lower completion time than label layouts produced with the *M0* and *P3* labeling methods, there is no significant difference between the completion times of the *M1*, *P1*, *P2* and *P3* labeling methods.

The comparison of completion times for all models evaluated together shows that the label layouts produced with the *M1*, *P1*, *P2*, and *P3* labeling methods provide a significantly lower completion times than label layouts produced with the *M0* method. There is no significant difference between the completion times of the *M1*, *P1*, *P2*, and *P3* labeling methods.

The results show that the *M0* labeling method produces ambiguous results for complex ghosted views where more than two semi-transparent objects overlap (drill, fork, and head models). The *M1* labeling method produces significantly better label layouts than the *M0* labeling method.

For simple ghosted views for which the *M0* labeling method produces mostly unambiguous results (digestive), the *M1* labeling method also produces unambiguous results. In this case label layouts produced both with the *M0* and *M1* labeling methods are comparable with label layouts created by our three experts.

If we evaluate all models together, our three experts *P1*, *P2* and *P3* can produce significantly better label layouts than the labeling method *M0*. Further, they are still able to produce slightly better label layouts than the labeling method *M1*, but not significantly better. Please recall that none of the experts is a professional illustrator.

In general, the labeling produced with the *M1* labeling method mediates the interconnection between textual and graphical information better than labeling produced with the *M0* labeling method. Therefore, the proposed labeling method *M1* should be preferred over the labeling method *M0* for semi-transparent 3D models.

When the experts created their label layouts, we observed one additional property of the label layout that they considered. All of them used symmetry in their label layouts for the symmetrical parts of the fork model, see Figure 13. When we asked them about this after the label layout was created, they confirmed that it was intentional. Our method *M1* does not consider any symmetry of the 3D objects. Although results from our evaluation do not suggest that the symmetry is affecting the function of the layout, this can be because the layout produced by our method *M1* is fairly symmetrical even though the symmetry of the 3D objects is not considered. Therefore, detailed evaluation of the impact of symmetry on the function of the label layout is needed. Further, we have observed that the experts were using different weights of the criteria for different labels. On the other hand, our proposed method uses the same criteria for all labels. The automatic calculation/estimation of the criteria weights for the individual labels may further improve the label layout.

6 Conclusion

We have presented a novel algorithm for the external labeling of ghosted views produced from moderately complex 3D models. The algorithm uses multiple criteria decision making to optimize the positions of anchors, leader lines, and label boxes. The optimization considers multiple, possibly contradicting, criteria which are solved using fuzzy logic. The algorithm operates in real-time. That allows the studying of the model and its labels from different views in an interactive session. We have conducted the evaluation with users to assess the influence of the introduced salience criteria on the ambiguity of the resulting labeling. The results of the evaluation show that significantly better label layouts are produced by the proposed method for complex ghosted views where more than two semi-transparent objects overlap. The label layouts produced with the proposed method are comparable to label layouts created by three experts recruited from user interface design and graphics design fields. Our evaluation thus indicates that the method can be used to automatically produce labeling comparable to that produced by humans who are well informed about the topic, but not professional illustrators. In our future work, we plan to evaluate how the method compares to the results produced by professional illustrators.

Acknowledgments

This research has been supported by the MSMT under the identification code 7AMB17AT021 within the activity MOBILITY (MSMT-539/2017-1).

References

[1] A. Agresti and B. A. Coull. Approximate is better than "exact" for interval estimation of binomial proportions. *The American Statistician*, 52(2):119–126, 1998.

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

[3] M. A. Bekos, M. Kaufmann, K. Potika, and A. Symvonis. Polygon labelling of minimum leader length. In *Proc. APVis*, pages 15–21, Australian Computer Society, Darlinghurst, Australia, 2006.

[4] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry*, 36(3):215–236, 2007.

[5] R. E. Bellman and L. A. Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17(4):B–141–164, 1970.

[6] M. Benkert, H. Haverkort, M. Kroll, and M. Nöllenburg. Algorithms for multi-criteria one-sided boundary labeling, In *Graph Drawing 2007, Revised Papers*, pages 243–254, Springer, Berlin, Germany, 2008.

[7] E. Bertini, M. Rigamonti, and D. Lalanne. Extended excentric labeling. *Computer Graphics Forum*, 28(3):927–934, 2009.

[8] S. Bruckner, S. Grimm, A. Kanitsar, and M. E. Gröller. Illustrative context-preserving exploration of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1559–1569, 2006.

[9] S. Bruckner, P. Rautek, I. Viola, M. Roberts, M. C. Sousa, and M. E. Gröller. Hybrid visibility compositing and masking for illustrative rendering. *Computers & Graphics*, 34(4):361–369, 2010.

[10] L. Čmolík and J. Bittner. Layout-aware optimization for interactive labeling of 3D models. *Computers & Graphics*, 34(4):378–387, 2010.

[11] F. de Moura Pinto and C. M. D. S. Freitas. Illustrating volume data sets and layered models with importance-aware composition. *The Visual Computer*, 27(10):875–886, 2011.

[12] J. Diepstraten, D. Weiskopf, and T. Ertl. Transparency in interactive technical illustrations. *Computer Graphics Forum*, 21(3):317–326, 2002.

[13] C. Everitt. Interactive order-independent transparency. *Technical report*, NVIDIA Corporation, 2001. http://developer.nvidia.com/docs/IO/1316/ATT/order_independent_transparency.pdf.

[14] J.-D. Fekete and C. Plaisant. Excentric labeling: Dynamic neighborhood labeling for data visualization. In *CHI '99*, pages 512–519. ACM, New York, USA, 1999.

[15] M. Fink, J.-H. Haunert, A. Schulz, J. Spoerhase, and A. Wolff. Algorithms for labeling focus regions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2583–2592, 2012.

[16] A. Gemsa, J.-H. Haunert, and M. Nöllenburg. Boundary-labeling algorithms for panorama images. In *Proc. of International Conference on Advances in Geographic Information Systems*, pages 289–298. ACM, New York, USA, 2011.

[17] T. Götzelmann, K. Hartmann, and T. Strothotte. Agent-based annotation of interactive 3D visualizations. In *Smart Graphics*, volume 4073 of *LNCS*, pages 24–35. Springer, Berlin, Germany, 2006.

[18] T. Götzelmann, K. Hartmann, and T. Strothotte. Contextual grouping of labels. In *Proc. of SimVis*, pages 245–258. SCS Publishing House, Magdeburg, Germany, 2006.

[19] T. Götzelmann, K. Hartmann, and T. Strothotte. Annotation of animated 3D objects. In *Proc. of SimVis*, pages 209–222. SCS Publishing House, Magdeburg, Germany, 2007.

[20] K. Hartmann, K. Ali, and T. Strothotte. Floating labels: Applying dynamic potential fields for label layout. In *Smart Graphics*, volume 3031 of *LNCS*, pages 101–113. Springer, Berlin, Germany, 2004.

[21] K. Hartmann, T. Götzelmann, K. Ali, and T. Strothotte. Metrics for functional and aesthetic label layouts. In *Smart Graphics*, volume 3638 of *LNCS*, pages 115–126. Springer, Berlin, Germany, 2005.

[22] J. Hensley, T. Scheuermann, G. Coombe, M. Singh, and A. Lastra. Fast summed-area table generation and its applications. *Computer Graphics Forum*, 24(3):547–555, 2005.

[23] Z.-D. Huang, S.-H. Poon, and C.-C. Lin. Boundary labeling with flexible label positions. In *Algorithms and Computation*, volume 8344 of *LNCS*, pages 44–55. Springer, Berlin, Germany, 2014.

[24] J. Krüger, J. Schneider, and R. Westermann. Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):941–948, 2006.

[25] M. Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[26] J. R. Lewis and J. Sauro. When 100% really isn't 100%: Improving the accuracy of small-sample estimates of completion rates. *Journal of Usability studies*, 1(3):136–150, 2006.

[27] M. Luboschik, H. Schumann, and H. Cords. Particle-based labeling: Fast point-feature labeling without obscuring other visual features. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1237–1244, 2008.

[28] K. Mühler and B. Preim. Automatic textual annotation for surgical planning. In *Proc. of Vision, Modeling, and Visualization Workshop*, pages 277–284, Braunschweig, Germany, 2009.

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

[30] M. Nienhaus and J. Döllner. Blueprints: Illustrating architecture and technical parts using hardware-accelerated non-photorealistic rendering. In *Proc. of Graphics Interface*, pages 49–56. Canadian Human-Computer Communications Society, Waterloo, Canada, 2004.

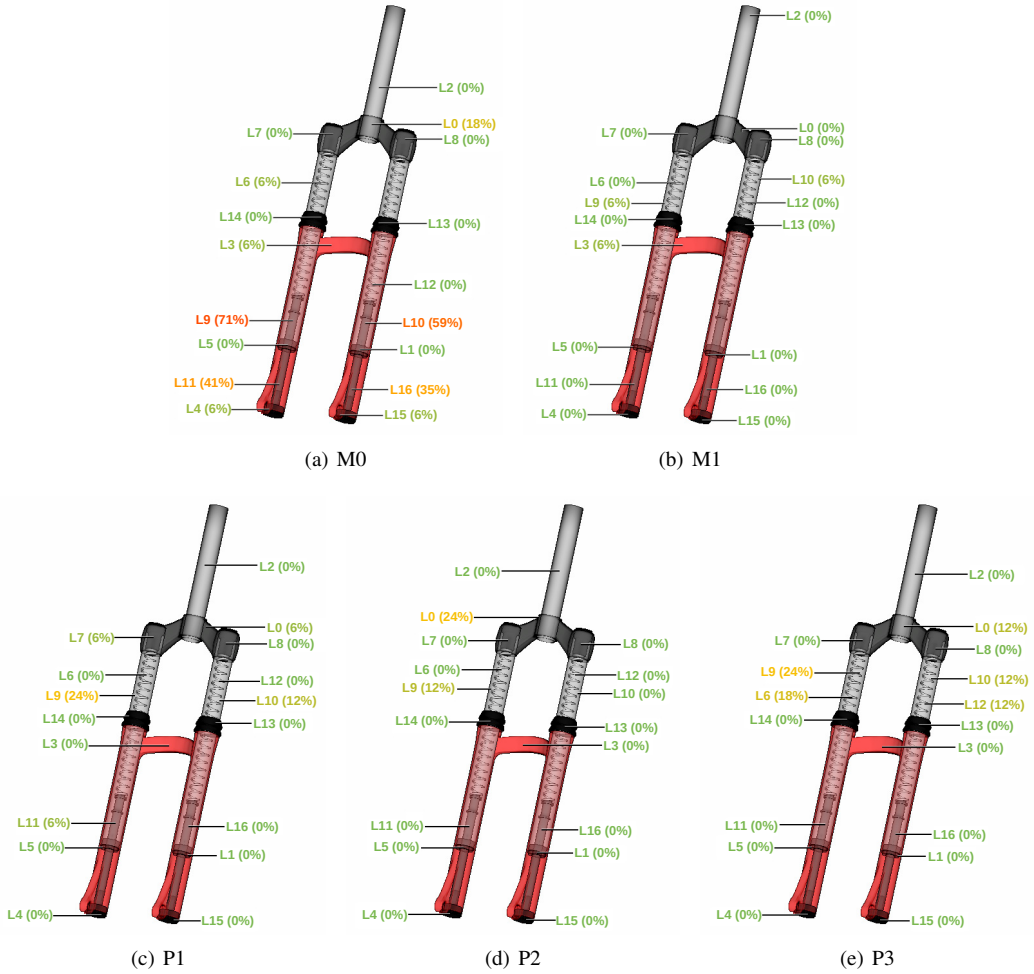


Fig. 13. The evaluated label layouts generated with the $M0$, $M1$, $P1$, $P2$, and $P3$ methods for the representative view of the *bike wheel fork* 3D model. The labels are color coded according to the error rate and the error rate is appended to each label.

[31] S. Oeltze-Jafra and B. Preim. Survey of labeling techniques in medical visualizations. In *Proc. of Eurographics Workshop on Visual Computing for Biology and Medicine*, pages 199–208. EG, Geneve, Switzerland, 2014.

[32] 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 CompuGraphics*, pages 201–210. GRASP, Queluz, Portugal, 1995.

[33] G. Rong and T.-S. Tan. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In *Proc. of Interactive 3D Graphics and Games*, pages 109–116. ACM, New York, USA, 2006.

[34] J. Sauro and J. R. Lewis. Estimating completion rates from small samples using binomial confidence intervals: comparisons and recommendations. In *Proc. of the human factors and ergonomics society annual meeting*, volume 49, pages 2100–2103. SAGE Publications, Los Angeles, USA, 2005.

[35] J. Sauro and J. R. Lewis. *Quantifying the user experience: Practical statistics for user research*. Elsevier, 2012.

[36] T. Stein and X. Décoret. Dynamic label placement for improved interactive exploration. In *Proc. of NPAR*, pages 15–21. ACM, New York, USA, 2008.

[37] M. Tatzgern, D. Kalkofen, D. Grasset, and D. Schmalstieg. Hedgehog labeling: View management techniques for external labels in 3D space. In *Proc. of IEEE Virtual Reality*, pages 27–32. IEEE, New York, USA, 2014.

[38] I. Vollick, D. Vogel, M. Agrawala, and A. Hertzmann. Specifying label layout style by example. In *Proc. of UIST*, pages 221–230. ACM, New York, USA, 2007.

[39] J. C. Yang, J. Hensley, H. Grün, and N. Thibieroz. Real-time concurrent linked list construction on the GPU. *Computer Graphics Forum*, 29(4):1297–1304, 2010.

[40] H. J. Zimmermann. Description and optimization of fuzzy systems. *International Journal of General Systems*, 2(1):209–215, 1975.

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

Jiří Bittner is an associate professor at the Department of Computer Graphics and Interaction of the Czech Technical University in Prague. He received his Ph.D. in 2003 from the same institution. For several years he worked as a researcher at the Vienna University of Technology. His research interests include visibility computations, real-time rendering, spatial data structures, and global illumination.