Adaptive Real-Time External Labeling for Interactive Visualization

Shan Liu and Yuzhong Shen
Department of Electrical & Computer Engineering, Old Dominion University
Norfolk, VA
sliu004@odu.edu, yshen@odu.edu

ABSTRACT

This paper presents an adaptive real-time external labeling algorithm for automatically placing labels on 3D models essential for creating annotation and visualization in virtual environments. The proposed approach considers a set of constraints for the label placement, which include but are not limited to, no overlapping labels, no intersections of leader lines with other leader lines or labels, short leader line lengths, and nearly uniform spatial distribution of labels. Screen space coordinates are utilized to compute the label positions that comply with the constraints. The 3D object coordinates are first converted to the camera coordinates for computing their projections on the viewing plane. The projected model is analyzed to determine the model contour and anchor points. The projection screen is evenly divided into four partitions, and the number of anchor points in each partition is adjusted so that the label density in each partition is relatively uniform and comparable. Then, label overlaps and leader line intersections are detected and eliminated, and the leader line length is adjusted using an iterative process. Finally, the labels are rendered on the screen. This approach operates in real-time by updating the label positions on the fly when the viewer changes viewing positions or directions. The proposed method has been implemented using the Unity game engine to demonstrate its ability to place labels automatically in real-time. Experimental results show that the proposed method produces high-quality label placement, thus providing practical utility for various interactive applications that involve virtual environments and having great potential for future Virtual Reality (VR) and Augmented Reality (AR) applications that require interactive annotations.

ABOUT THE AUTHORS

Shan Liu is a Ph.D. student in the Department of Electrical & Computer Engineering (ECE) at Old Dominion University. Her research interests are Virtual Reality and Augmented Reality, especially in the 3D digital instruction manual.

Yuzhong Shen received his BS degree in Electrical Engineering from Fudan University, Shanghai, China, MS degree in Computer Engineering from Mississippi State University, Starkville, Mississippi, and Ph.D. degree in Electrical Engineering from the University of Delaware, Newark, Delaware. His research interests include virtual Reality, Augmented Reality, visualization and computer graphics, transportation modeling and simulation, general modeling and simulation, and signal and image processing.

Dr. Shen is Professor of the Department of Electrical and Computer Engineering at Old Dominion University. Prior to joining Old Dominion University, Dr. Shen worked as an Engineer and a Senior Engineer at Weifang Hua-Guang Technologies, China, as a Research Assistant at National Science Foundation Engineering Research Center for Computational Field Simulation at Mississippi State University, as a Research Assistant at the Department of Electrical and Computer Engineering at the University of Delaware, and as a Senior Research Scientist at Virginia Modeling, Analysis, and Simulation Center (VMASC) at Old Dominion University. Dr. Shen is a Senior Member of IEEE.

Adaptive Real-Time External Labeling for Interactive Visualization

Shan Liu and Yuzhong Shen
Department of Electrical & Computer Engineering, Old Dominion University
Norfolk, VA
sliu004@odu.edu, yshen@odu.edu

INTRODUCTION

From education to medicine to manufacturing, one essential task is understanding the details of complex objects, such as assembled machinery or a human anatomy model. Well-placed informative labels that connect visual and textual information can provide accurate descriptions and instructions for objects of interest. However, traditional printed product manuals are difficult to follow when users interact with complex physical objects containing multiple components. Users are often frustrated by the manual's lack of clarity, e.g., the actual parts do not visually match the visual illustrations or textual description in the product manual.

Virtual environments can effectively address the issues in traditional printed product manuals or instructions by annotating virtual object components in real-time. In virtual environments, labels provide additional information about the object in the real world. However, how to automatically annotate various parts of objects from multiple perspectives in a virtual environment in real-time remains an open problem.

Many automatic labeling methods have been developed to mimic classic hand-drawn illustration annotation styles (Ali, Hartmann, and Strothotte 2005; Bekos, Niedermann, and Nöllenburg 2019; Cmolık et al. 2020). There are two main labeling algorithms for object annotations: internal and external. Internal labeling algorithms generate labels that overlay visual objects. In contrast, external labeling algorithms create labels outside the objects and are connected to the labeled objects by leader lines to avoid occlusions and confusion.

Due to users' constantly changing views in virtual environments, the label layout must be adaptively updated to resolve constraint violations, making it easy for users to keep track of the object components of interest. Aside from creating static layouts of annotations for a single projected object, labeling algorithms also need to maintain consistency when the viewpoint changes.

This paper proposes an adaptive algorithm for the external labeling of point features that adapts to different object views. The main contribution lies in applying the idea of real-time external labeling to interactive virtual environments. The paper discusses methodologies for creating easily legible, well-spaced, and temporally coherent label layouts for complex objects when the perspective changes in real-time. These methodologies reduce cognitive load and improve cognitive efficiency in the visualization during viewpoint changes.

RELATED WORK

External labeling algorithms have been investigated in computer science from practical and theoretical points of view and have been applied in many fields over the last twenty years. While internal labeling has achieved great success in applications such as maps, external labeling is especially valuable for applications such as annotating object parts in a virtual environment, as external labels do not interfere with other objects. The application of external labeling algorithms can be divided into static and dynamic applications based on whether labels change over time (Bekos, Niedermann, and Nöllenburg 2019).

There have been many examples of external labeling in static applications, such as atlases of human anatomy (Niedermann, Nöllenburg, and Rutte 2017) and visual dictionaries (Cmolık and Bittner 2019). However, this type of research only focuses on a fixed viewing specification or restricts the objects to be annotated, which cannot meet users' needs to observe objects in a virtual environment in real time.

External labeling in dynamic applications applies external labeling in digital visualizations that change over time. This labeling algorithm is usually found in view management systems, where users can interactively explore 3D models (Madsen et al. 2016). Most current dynamic labeling algorithms focus on solving the label flickering and jumping problem. Ali et al. presented an algorithm that considers layout decisions from the previous frame to achieve a frame-coherent label layout during user interactions (Ali, Hartmann, and Strothotte 2005). However, the random changing range of objects in an interactive virtual environment makes it impossible to solve the flicking and jumping problem by inheriting the data from the previous frame, which may lead to label overlapping. This paper also inherits the data from the previous viewpoint to maintain temporal coherence but simultaneously update label positions to satisfy labeling constraints, such as no label overlap. Tatzgern et al. proposed placing labels in 3D object space and using geometric constraints to achieve the required label layout and behaving consistently over time during viewpoint changes (Tatzgern et al. 2014). However, this method requires the user to set up multiple planes at runtime to solve the problem of label overlap, which increases computational costs and cannot fully solve the overlapping problem between the planes. We also employ geometric constraints to obtain a satisfactory label layout in real-time, but only in 2D projection space, to reduce computational costs and increase the speed of the labeling algorithm.

TERMINOLOGY AND CRITERIA

Labels represent textual or symbolic descriptions of the model's parts. Geometrically, the label is defined as the axis-aligned rectangle containing the attached information of the feature in this paper. An illustration with external labeling consists of a projected object, a set of labels outside the object, and their corresponding leader lines. Leader lines are the line segments connecting the labels with their parts. Only straight-line segments are used as leader lines in this paper. One endpoint of the leader line is restricted to be the center point of the labeled part in this paper. The other endpoint of the leader line, named as the reference point in this paper, is a point on the boundary of the label, which is restricted to one corner or the midpoint of one particular edge of the label.

In order to formalize the criteria for our algorithm, we utilize well-established labeling guidelines for external labeling of point features found in the literature (Bekos, Niedermann, and Nöllenburg 2019). The set of rules adapted to the needs of external dynamical labeling is as follows:

R1. The label should be placed outside at some distance from the model component.
R2. Labels are placed horizontally.
R3. Labels must not overlap.
R4. There are no intersections of leader lines with other leader lines or labels.
R5. The leader lines have small lengths.
R6. The labels are distributed evenly on the screen.

ALGORITHM CORE

The proposed method is a screen-space technique operating in a view plane on which the 3D models are projected. Figure 1 presents the system architecture of the approach. The content presented on labels is obtained from the product manuals. The projected model is analyzed to determine the model contour and anchor points used to satisfy rule R1. The projection screen is divided into four partitions, and the number of anchor points in each partition is adjusted so that the density of each partition is relatively consistent with satisfying rule R6. Then, label overlaps and leader line intersections are detected and eliminated, and the leader line length is adjusted as short as possible simultaneously. Finally, the labels are rendered on the top of the screen.

The most vital part of the algorithm is Label Layout which includes three important actions: label density adjustment, label rotation, and leader line extension. Label density adjustment is used to satisfy rule R6. Label rotation and leader line extension are combined or used independently in Local Label Placement and Global Label Refinement procedures to satisfy the rules R3, R4, and R5.

Label Density Adjustment

Label density refers to the number of labels per unit area. This paper calculates a rough estimation of label density as the number of labels in a screen partition. The partition of the screen is shown in Figure 2. The screen is divided into four partitions: 0, 1, 2, and 3. In order to simplify the formulation, each partition includes a start boundary and an end boundary, and this paper uses a clockwise order for all labels. The anchor points of the model are divided into four groups based on their coordinates on the screen. Since the number of anchor points in each partition is not necessarily the same, adjusting the number of anchor points in each partition is essential to distribute labels evenly before adjusting the label position.

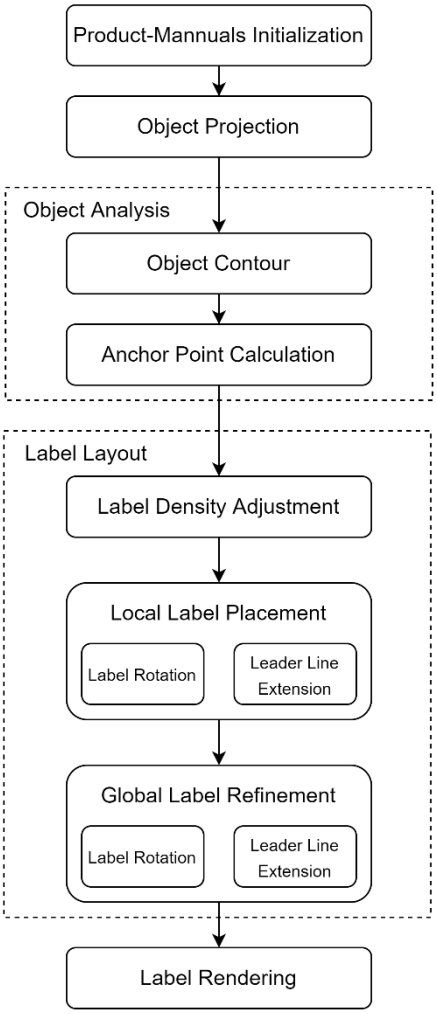


Figure 1. The system architecture of the proposed labeling algorithm.

Label density adjustment occurs when two adjacent partitions' numbers of anchor points differ significantly. If the difference between the number of anchors in two adjacent partitions is more than one and the larger of the two numbers is more than three, the algorithm will perform label density adjustment. First, the number of anchor points to be adjusted is determined by calculating the average number of anchor points in the two partitions. Then the distance between each anchor point and the shared boundary of the two partitions is calculated and sorted. The anchor point that is the shortest from the shared boundary of the two partitions is then assigned to the partition with fewer anchors. The labeling density of each partition is adjusted in pairs to shorten the leader line length as much as possible, as shown in Figure 2.

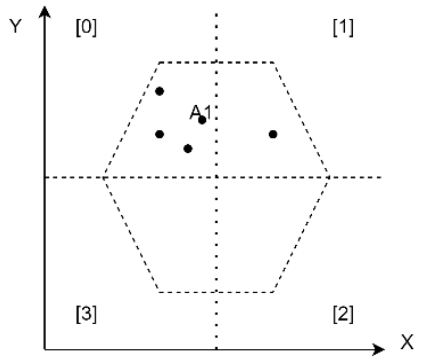


Figure 2. The schematic of screen partition and label density. After label density adjustment, the partition attribute of anchor point A1 is adjusted from Partition 0 to Partition 1. In other words, the label for anchor point A1 is placed in Partition 1.

Label Rotation

Following the determination of the number of anchor points for each partition, the shortest distance from each anchor point to the model contour in the partition is calculated, and the intersection with a particular segment in the section's contour found by the shortest distance is the initial value of the reference point. Therefore, the initial reference points summarize all the leader lines with the shortest length. Furthermore, its corresponding label's position is determined based on the angle between the leader line and the positive X-axis, as illustrated in Figure 3.

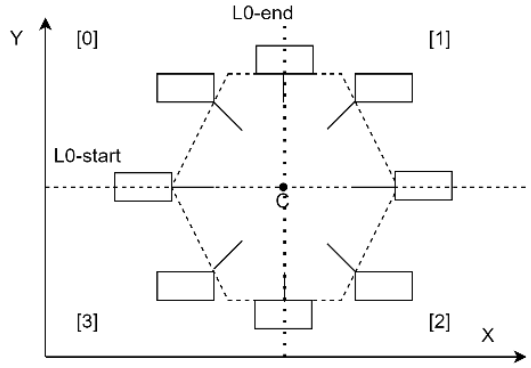


Figure 3. The schematic of the relationship among the label, the reference point, and the leader line. The angle between the leader line and the positive X-axis determines the position of the label to the reference point.

Label rotation is one of the primary adjustment methods in this paper. When two labels overlap, or one of the label's leader lines intersects with another leader line or another label, one label is selected to rotate around its anchor point clockwise or counterclockwise to solve these issues. Let the chosen leader line rotate a certain angle around its anchor point, extend or shorten the leader line to intersect with the model contour, get a new reference point, and then find the new label position according to the angle between the leader line and the positive horizontal axis.

Leader Line Extension

The leader line extension is the other one of the primary adjustment methods in this paper. When the label density in a specific partition is so high that label rotation cannot avoid label overlap, leader line intersection, or leader line and label crossing, leader line extension is an effective way to solve the above problems. What needs to be paid attention to when extending the leader line is to comply with rule R5; the length of the leader line is as short as possible.

ALGORITHM ENGINEERING

It is necessary to determine the relationship of one or more labels to other labels. Moreover, it is necessary to rearrange some labels to avoid leader line intersection, label overlap, and the intersection between leader line and label. Initial experiments showed that a naïve implementation of the above adjustment methods does not yield acceptable labeling results. This section further describes how these methods can be implemented efficiently to solve labeling problems.

Local Label Placement

This paper uses the sort-and-sweep method (Ericson 2005) to sort all the labels in a particular partition and then detect whether there are intercrosses between the leader lines, overlaps among the labels, and intersections between one label and the other label's leader line. The labels are sorted clockwise based on the angle between the start boundary of the partition and the line connecting the reference point and the screen center point. For example, as shown in Figure 4, the start boundary of partition 1 is L1-start, and the end boundary is L1-end. There are n labels. These labels are sorted clockwise.

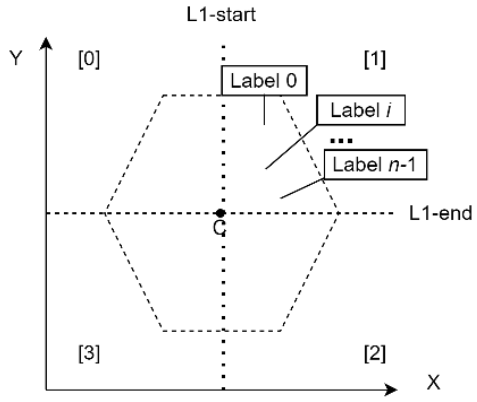


Figure 4. The schematic diagram of labels that are arranged in a clockwise direction in Partition 1.

In a particular partition, two adjacent labels, including the current and previous labels, are adjusted as a group. Therefore, the labels in a specific partition are divided into three types: the partition's first label, the second includes labels from the second to the penultimate, and the third is the last. The pseudo-code of the Local Label Placement procedure is shown in Figure 5.

Since the first label in the partition is the current label and has no previous label, it is processed based on its relationship with the start boundary.

1. If the first label crosses the start boundary, it is rotated clockwise by a certain number of degrees $\varepsilon_i = \theta_i/\alpha$, $\theta_i = 90/\Sigma_i$, $i = 0, \ldots, 3$, α is an adaptive parameter, set as 3 in this paper, Σ_i is the number of labels in partition i. Then the new label position is calculated, and this step is repeated until the first label is located inside the partition.

2. If the first label does not cross the start boundary, the algorithm calculates two distances: 1) the average distance of the remaining reference points, dl_1, which is the distance from the current reference point to the contour endpoint in the current partition divided by the number of anchors less one, and 2) the average distance of all reference points, dl_0, which is the distance from the contour start point to the contour endpoint in the current partition divided by the number of anchors. If dl_1 is less than dl_0, the current label is rotated counterclockwise until dl_1 is equal to or greater than dl_0, or the current label is about to exceed the partition's start boundary.

```
for each partition {
    Detect the type of the current label
    if (the first label) {
        Detect if the label is within the start boundary, and do label rotation to fit the
      label in the bounds if necessary;
        Calculate two average distances, dl0 and dl1;
        Do label rotation if necessary to let dl1 >= dl0.
    }
    else if (the second type of label) {
        Detects if leader line intersection for adjacent labels and swaps label's
      reference point positions if necessary;
        Calculate the rotation angle of the leader line relative to the initial position
      and compare it with the threshold value to determine whether to do label rotation
      or leader line extension;
        Calculate two average distances, dl0 and dl1;
        Do label rotation if necessary to let dl1 >= dl0.
    }
    else {
        Detects if leader line intersection for adjacent labels and swaps label's
      reference point positions if necessary;
        Detect if the label is within the end boundary, and do label rotation or leader
      line extension to fit the label in the bounds if necessary.
    }
}
```

Figure 5. The pseudo-code of local label placement.

For any label in the second type, its group includes two labels. The first step is to check whether the leader lines of the two adjacent labels intersect. If so, the positions of the reference points corresponding to the crossed leader lines are exchanged. Then, the sort-and-sweep algorithm is used to re-sort the reference points on the same partition clockwise around the model's contour and detect whether there is still a leader line crossing until there is no leader line crossing.

The second step is to detect whether the current label overlaps with the previous label or intersects with the leader line of the previous label. The current label is rotated clockwise if they overlap or intersect. And then, the label position is updated. Furthermore, this step is repeated until the difference between the angle from the new leader line to the positive X-axis and the angle between the initial leader line and the positive X-axis is larger than the threshold, θ_i, or there is no label overlap and no leader line intersection in the group. If the angle between the new leader line and the initial leader line exceeds the threshold θ_i, and label overlap or leader line intersection still persists, the current leader line is extended by a length step size $\mu = \rho \times l_c$, ρ is an adjustable parameter, set as 0.01 in this paper, and l_c is the length of the current leader line until there is no label overlap and no leader line intersection in the group, or the leader line length is more than γ times the initial length, γ is an adjustable parameter, set as 1.2 in this paper, or the label is about to extend beyond the screen boundaries. If the current label does not overlap nor intersect the previous leader line, the average distance dl_1 and dl_0 are calculated as discussed before. Similarly, if dl_1 is less than dl_0, the current leader line is rotated counterclockwise until dl_1 is equal to or more than dl_0 or the current label is about to overlap the previous label or intersect with the previous leader line.

Like the second type of label, the last label in the current partition is first checked to check whether it intersects with the previous leader line. If it does, a similar exchange process is performed. Then the last label is checked to see if it overlaps with the previous label. If it overlaps, its relationship with the end boundary of the partition is further examined. If the last label does not exceed the end boundary, the last label is rotated clockwise by ε_i until the last label does not overlap the previous label or is about to exceed the end boundary. However, if the last label exceeds the end boundary but still overlaps the previous label, the last leader line is extended by μ until the last label does not overlap its previous label or is about to exceed the screen boundaries.

Global Label Refinement

After adjusting the label positions in each partition, it is still necessary to examine whether adjacent labels in adjacent partitions overlap and whether their leader lines cross each other. The pseudo-code of the Global Label Refinement procedure is shown in Figure 6.

```
Detect if leader line intersection for adjacent labels and swap labels' reference points if
necessary;
Detect the distance between adjacent leader lines and swap labels' reference points if
necessary;
Recursive refinement {
    Detect if labels overlap, leader lines intersect, or leader lines intersect with other
labels;
    Calculate two distances between labels of two groups of adjacent labels, dist_cp and
dist_cn;
    Compare two distances to determine label rotation or leader line extension for which
group.
}
```

Figure 6. The pseudo-code of global label refinement.

Like Section Local Label Placement, the sort-and-sweep method is first used to detect whether the leader lines of adjacent labels cross. If the leader line intersection is detected, their reference points are directly swapped, and all the reference points and labels are updated accordingly.

The next step is to enlarge the distance between two adjacent close leader lines. If the distance between two adjacent leader lines is less than a threshold (set as half the label's height in this paper), the steps below are followed to enlarge the distance. First, their reference points are exchanged, the new distance between two adjacent leader lines is calculated if there is no label overlap or leader line cross, and the new distance is compared with the origin distance. If the new distance is larger, the labels are updated. Otherwise, keep it as it is.

The final step deals with label overlap and crossing between one leader line and its adjacent label using recursive refinement. If the current label overlaps the next label or crosses the following leader line, the distances between the current and the previous label and between the current and the next label, $dist_{cp}$ and $dist_{cn}$, are calculated to determine which label is adjusted. If the previous label is farther from the current label, the current leader line is rotated counterclockwise. The current reference point and the current label are updated accordingly. If there is an overlap or intersection between the updated current label and the previous label, the leader line of the previous label is rotated counterclockwise, and the previous label is updated. The above steps are repeated until the current label does not overlap or intersect with the previous and the next label. Otherwise, if the next label is further from the current label, the next label is rotated clockwise, and the next reference point and label position are updated accordingly.

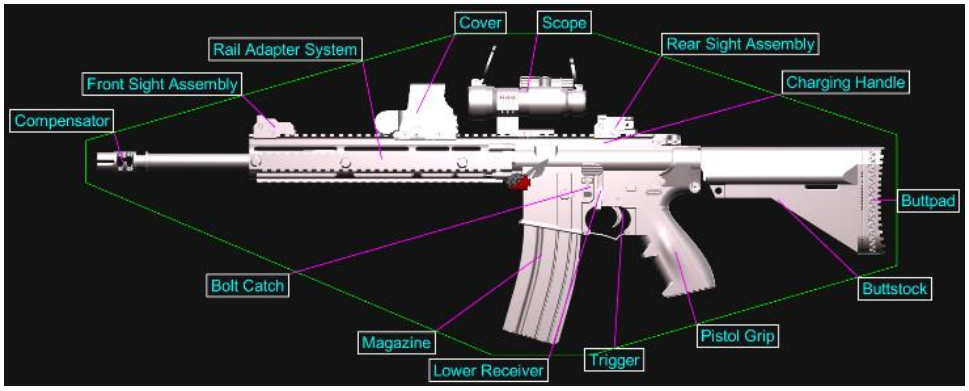
The steps in these two sections, Local Label Placement and Global Label Refinement, are repeated until there is no label overlap and leader line cross, or it is up to the maximum iteration number, set as 150 in this paper.

Viewpoint Change Adaptation

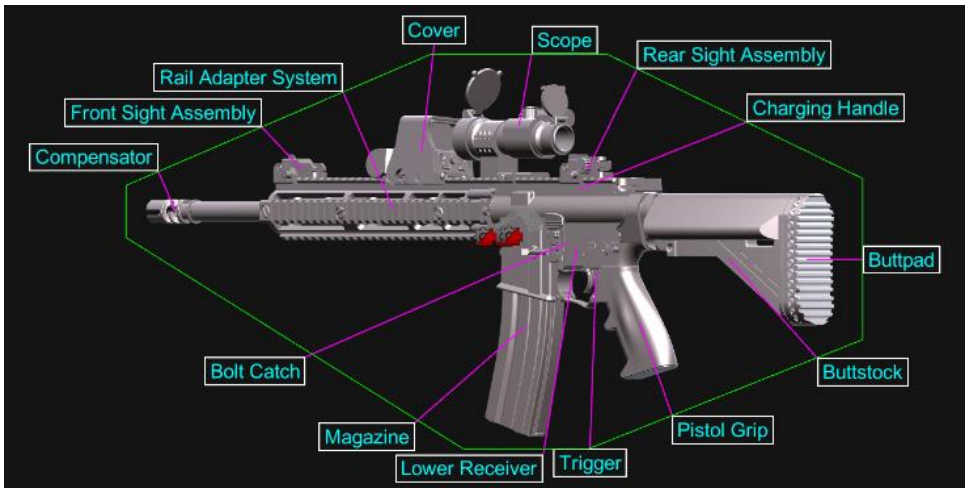
In order to maintain the temporal coherence of the labels when the viewpoint changes, the proposed algorithm initializes the label positions with the label positions before the viewpoint change when the partition attributes of the anchor points do not change and then refines the label positions using the approaches discussed in previous sections. The approach produces good results when the viewpoint changes are gradual and smooth, which accounts for most of the use scenarios.

EXPERIMENT RESULTS

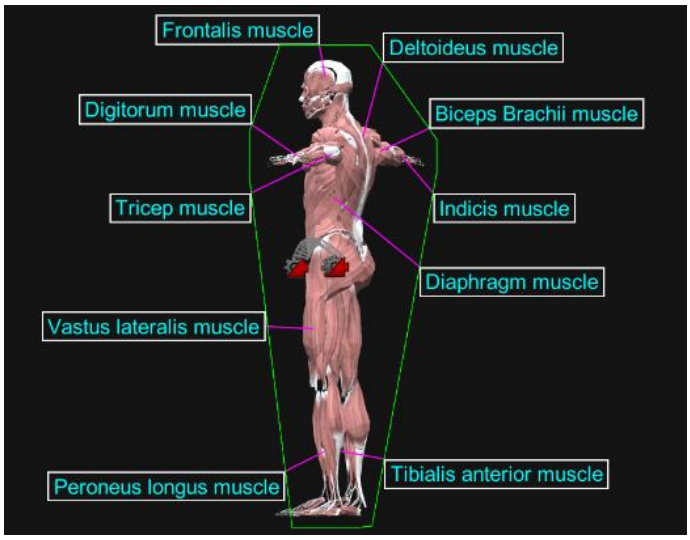
The proposed approach has been implemented in the Unity game engine to demonstrate its ability to place labels automatically in real time when the viewpoint changes. Figure 7(a) illustrates the effect of label density adjustment on optimizing label distribution when anchors are unevenly distributed on the screen. Figure 7(b) demonstrates the temporal coherence of labels when the viewpoint changes. Figure 7(c) shows the results of the proposed method applied to an anatomical model. Figure 7 shows the performance of this method under multiple models and different viewpoints.



(a)



(b)



(c)

Figure 7. Label placement with the proposed method. (a) The label layout for the Rifle model. (b) The temporally coherent layout for the Rifle model when the viewpoint changes. (c) The label layout for an anatomical model.

CONCLUSIONS

This paper implements a point-feature annotation placement algorithm for the automatic placement of external labels for 3D models and has experimented with it in several objects. This approach fulfills the desired objectives of labeling algorithms (e.g., avoiding label overlapping) and behaves consistently over time during viewpoint changes. The implementation provides a smooth transition between changing models. The proposed method has excellent potential for future AR and VR applications that require interactive annotations.

REFERENCES

Ali, Kamran, Knut Hartmann, and Thomas Strothotte. 2005. "Label Layout for Interactive 3D Illustrations." *Journal of WSCG* 13 (1):1-8.

Bekos, Michael A., Benjamin Niedermann, and Martin Nöllenburg. 2019. "External Labeling Techniques: A Taxonomy and Survey." *Computer Graphics Forum* 38 (3):833-860. doi: 10.1111/cgf.13729.

Cmolık, Ladislav, and Jirı Bittner. 2019. "Real-Time External Labeling of Ghosted Views." *IEEE Transactions On Visualization and Computer Graphics* 25 (7):2458-2470. doi: 10.1109/TVCG.2018.2833479.

Cmolık, Ladislav, Vaclav Pavlovec, Hsiang-Yun Wu, and Martin Nollenburg. 2020. "Mixed Labeling: Integrating Internal and External Labels." *IEEE Transactions on Visualization and Computer Graphics* 28 (4):1848-1861.

Ericson, Christer. 2005. *Real-Time Collision Detection*. San Francisco, USA: Morgan Kaufmann Publishers.

Madsen, Jacob Boesen, Markus Tatzgern, Claus B. Madsen, Dieter Schmalstieg, and Denis Kalkofen. 2016. "Temporal Coherence Strategies for Augmented Reality Labeling." *IEEE Transactions on Visualization and Computer Graphics* 22 (4):1415-1423.

Niedermann, Benjamin, Martin Nöllenburg, and Ignaz Rutte. 2017. "Radial Contour Labeling with Straight Leaders." The 10th IEEE Pacific Visualization Symposium (PacificVis 2017) Seoul, Korea, April 18-21.

Tatzgern, Markus, Denis Kalkofen, Raphael Grasset, and Dieter Schmalstieg. 2014. "Hedgehog Labeling: View Management Techniques for External Labels in 3D Space." IEEE Virtual Reality (VR), March 29 - April 2.