Label Layout for Interactive 3D Illustrations

Kamran Ali, Knut Hartmann, and Thomas Strothotte
Department of Simulation and Graphics
Otto-von-Guericke University of Magdeburg
Universitätsplatz 2, D-39106 Magdeburg / Germany
{kamran, knut, tstr}@isg.cs.uni-magdeburg.de

ABSTRACT

Hand-made illustrations in scientific and technical textbooks commonly use internal and external labels or legends to establish co-referential relation between pictorial elements and textual expressions. By analyzing the most complex examples, we extracted several label layout styles and classified them. We propose a variety of real-time label layout algorithms that aim to produce nice and clean layouts. In order to achieve a frame-coherent label layout during user interactions, the algorithms consider layout decisions from previous frame. Moreover, several evaluation criteria to measure the quality of static as well as dynamic label layouts are presented.

Keywords

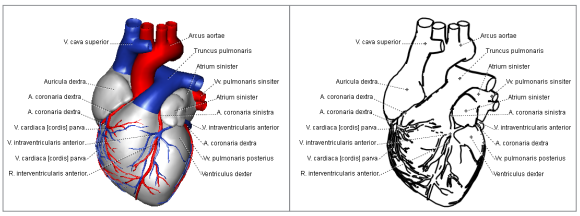
Label-Layout, External Labeling, Text-Image Integration, Multi-Modal Presentations

1 INTRODUCTION

Interactive tutoring systems aim at presenting information in the most effective way. The dual coding theory [CP86] suggests that humans posses two independent processing systems—one for visual and the other for verbal elements. Hence, using two channels, more material can be conveyed, but their content has to be integrated mentally.

Human illustrators employ a number of techniques to establish *co-referential* relation between visual and verbal elements. Labels, legends, and figure captions provide denotations, technical terms, and descriptions for visual elements. However, their automated integration within an interactive 3D environment remains a big challenge.

Text *labels* either overlay visual objects or placed outside (*internal* vs. *external* labels). *Connecting lines* reveal co-referring external labels and visual objects, whereas *anchor points* ease the identification of visual objects. In this work, the term *label layout* refers to the determination of the positions of anchor points and external labels, which are linked with connecting lines



using a specific line style. Moreover, the term *graphical model* refers to a complex visual object with separate individual visual objects.

We extracted requirements for several layout styles which are prevalent in hand-made illustrations and present techniques towards an automated generation of label layouts in real-time. In order to achieve a frame-coherent label layout during user interactions, these algorithms consider layout decisions from the previous frame. Moreover, the system facilitates automatically generated *legends* for graphical models and *textual explanations* for visual objects. The mental integration of information presented in 3D and legend viewer is aided through a synchronized object selection and highlighting mechanism.

The paper starts by giving a review of related work in Section 2. Section 3 states the requirements for dynamic labeling system. In Section 4 several label layout styles are classified. Section 5 presents the architecture of our label layout system and provides the algorithms to generate several layouts. Moreover, coherency aspects and the application of labels in legends are described. Section 6 states the *evaluation criteria* to measure the quality of layouts. Finally, Section 7 discusses directions of future research.

The Journal of WSCG, Vol. 13, ISSN 1213-6964
WSCG'2005, January 31-February 4, 2005
Plzen, Czech Republic.

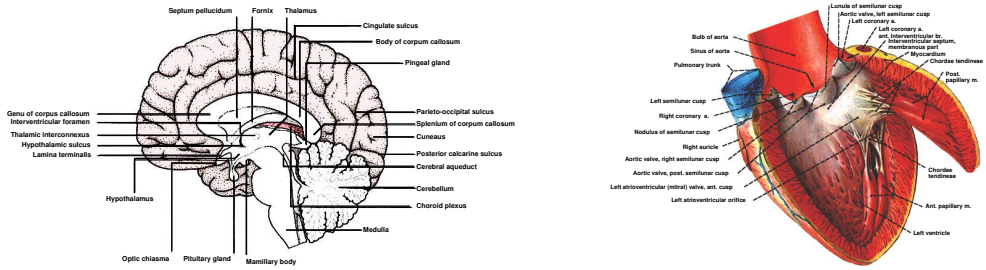


Figure 1: Variety in the layout styles (Source: [Rog92, p. 317] and [SPP97, p. 81]).

2 RELATED WORK

The label layout problem has received much attention in non-interactive cartographic applications [CMS95] where the labels have to be placed for point, line, and area features. However, the label placement is independent of the shape of graphical features and can be unified for all kinds of features [KT98]. Finding the optimal solution of the labeling problem (i.e., without overlapping labels) is proven to be NP-hard [MS91]. Therefore, several approximation methods have been developed to reduce the computational complexity.
A number of interactive multi-modal systems integrate external labels into the visualization of geometric objects. But most of them rely on fixed regions for visual and textual elements (e.g., [PRS97]) or a manual label layout (e.g., [RSHS03]). Only a few solutions have been proposed to integrate internal and/or external labels into interactive 3D applications (e.g., [BFH01]), but they lack the ability to generate a variety of layouts which are often seen in hand-made illustrations.

3 REQUIREMENTS

In dynamic environments an effective layout must fulfill a number of requirements ([Imh75, FP99]):

Readability: Labels must not overlap,
Unambiguity: Labels clearly refer to their objects,
Pleasing: Prevent visual clutter,
Real-Time: Compute layouts at interactive rates,
Frame-Coherency: Prevent visual discontinuities,
Compaction: Reduce the layout area.

These requirements may conflict with each other and with another demand: *label as many visual objects as possible.* Some of these requirements can be evaluated easily, whereas the extraction of criteria for the second and third aspect is less obvious. We use several heuristics to achieve a pleasant and unambiguous layout:

(i) Place anchor points over salient positions,
(ii) Place labels near to their corresponding objects,
(iii) Align labels mutually and with respect to the graphical objects, and
(iv) Eliminate line crossings.

The label layout algorithms are incorporated into an interactive application where visual discontinuities between subsequent frames must be avoided. Moreover, the layouts should be as compact as possible to fit on the limited screen space. Finally, the layout algorithms have to cope with situations where some labels do not fit into the given screen space. As the computation of an optimal solution is NP-hard, several simple yet effective methods are proposed that can be carried out in real-time.

4 LAYOUT STYLES

The material in this section is based on a manual analysis of label layouts in hand-drawn illustrations. For this purpose, we chose anatomic atlases, anatomic textbooks, and visual dictionaries because of their making extensive use of external labels and due to the extraordinary quality of their label layouts.
The manual analysis reveals that human illustrators use a number of different label layout styles with style-specific illustration techniques and properties (see Figure 1). Therefore, we classified them according to their common properties (see Figure 2):

Straight-Line: Labels and anchor points are connected with straight lines (see Figure 1-Right).
Orthogonal: Connecting lines are axis-aligned and the bends are made at orthogonal angles (see Figure 1-Left).
Flush Layout: Labels are assigned to distinct spatial areas (see Figure 3-a):

- *Flush Left-Right:* Labels are placed on the left and/or right side of the graphical model.
- *Flush Top-Bottom:* Labels are placed on the top and/or bottom of the graphical model.

Circular Layout: Labels are aligned on the silhouette of the graphical model in a circular fashion (see Figure 3-b):

- *Ring*: Labels are placed at regular intervals on a ring which encircles the graphical model.
- *Radial*: Labels are placed in radial form with respect to a common origin.
- *Silhouette-Based*: Labels are placed near the silhouette of graphical model at positions closest to their anchor points.

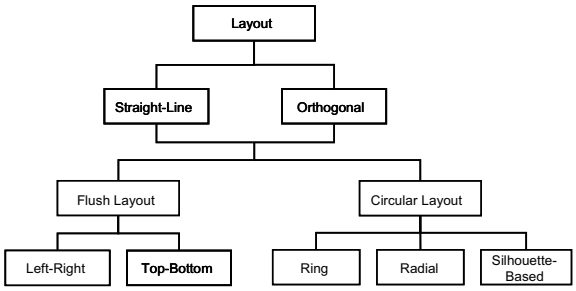


Figure 2: Layout Classification.

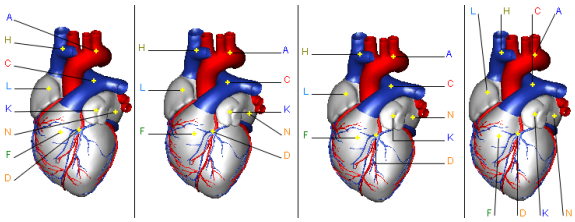
These styles are adopted in order to meet space requirements, to bring conformity in different illustrations, to maintain visual balance, and to ease reading. The most interesting observation is that there are some general but also several style-specific requirements.

5 AUTOMATED LABEL LAYOUT

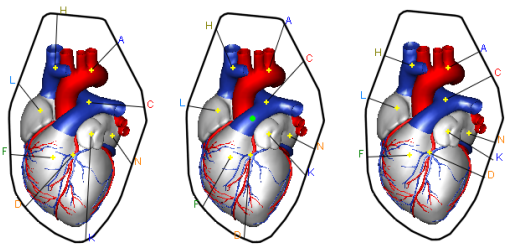
In this section, the properties and constraints of labels, anchor points, and connecting lines are defined. Moreover, we describe our approach towards the dynamic layout of label in interactive systems.

5.1 Object Properties and Constraints

The amount of text displayed in labels can range from one- or two-letter symbols to multi-line paragraphs. However, most often labels comprise few words on a single line. We classify labels into the following categories:



(a) *Flush* layouts combined with *straight-line* or *orthogonal* styles.



(b) *Ring*, *radial* and *silhouette-based* layouts.

Figure 3: Examples of various layout styles.

(i) *single-line* (max. 50 characters),
(ii) *multi-line* labels, or
(iii) *legend keys* (max. 2 characters).

For labels of different sizes, more constraints are needed to avoid label overlaps and line intersections. Thus, achieving a balanced layout becomes more problematic. Therefore, our layout strategy is restricted to single-line labels and legend keys which both have a fixed height and width. This constraint enables us to represent labels as *zero-sized points* and to maintain a minimal vertical and horizontal gap between them. Multi-line descriptions and legend text are provided on request. They do not alter their positions and have a semi-transparent background. User can pin them anywhere on the screen. The positions of all kinds of labels, anchor points, and connecting lines are specified in view-plane coordinates.
All objects are assigned *display priorities* (that consider projection size) and *user priorities*. For complex models, the labels can be filtered according to their *degree of interest*. If there is not enough place to display all label, objects with the smallest priorities are chosen and their labels are ignored.

5.2 System Architecture

Figure 4 presents an overview of our approach. The content presented in labels is provided by an external *domain expert*. The system works internally on 2D projection of 3D scene where individual visual objects are color-coded uniquely (*color-code image*). The rendered image is analyzed to determine visible objects and anchor points. Layout-specific algorithms determine initial positions for labels. Then label overlaps are eliminated and line intersections are resolved. If required, layout compaction is performed. Finally, the labels are rendered with chosen decoration style on top of the scene.

5.2.1 Domain Expert Initialization

The co-referential relation between textual annotations and visual objects is established by using an external knowledge base. When the system loads a 3D model, the domain expert defines a color-coding scheme for visual objects and provides textual descriptions.

5.2.2 Image Analysis

This module segments *color-code images*. For every rendered frame, it creates a list of all segments, their sizes, extents, and colors (to identify the visual objects). For each visible object, it determines one anchor point. Since an anchor point is intended to support the identification of visual object and its distinction from the remaining objects, its position is crucial to prevent co-referential mismatch. From observa-

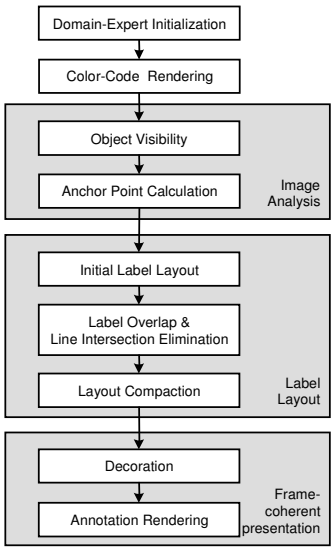


Figure 4: System architecture.

tions, we define the following heuristics to determine anchor points:

(a) They must overlay their corresponding objects.
(b) Place anchor points inside the biggest segments.
(c) Place them at the most internal locations of these segments.
(d) Avoid clusters of anchor points.

Anchor Point Calculation: If the objects have 'L' or 'U' shape, neither the center of the bounding box nor the centroid guarantee to fulfill the first condition. To compute the most internal pixel in a segment, we apply *distance function* on *color-coded images*. For every pixel this function computes its distance to the closest segment boundary and stores these values in a *distance image* (see Figure 5-Right).
There are several variants of this function which employ different metrics: *Euclidean*, *Manhattan*, and *Chessboard*.
The last two metrics are faster but less accurate. In order to reduce the computational expense of the euclidean metric, we adopt the *pseudo euclidean metric* d_{34} [AdB88][1]. The d_{34} metric assigns distance value 3 to horizontal and vertical neighboring pixels; the value 4 is assigned to diagonally connected pixels. We implemented a 2-pass algorithm to compute *distance image* [RP68] using the d_{34} metric. For each visual object a mask is placed over the *distance image* and the highest distance value is returned as anchor point.

Elimination of Anchor Point Clusters: In order to avoid referential mismatches or ambiguities, anchor points should not form clusters. Therefore, *repulsive*

[1]An approximation which purely uses integer operations and avoids square roots.

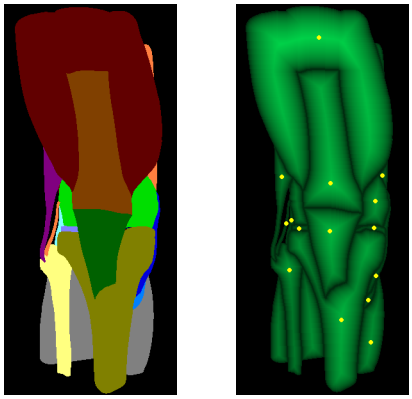


Figure 5: *Color-code image* (left) and *distance image* (right) with overlaid anchor points.

forces aim at separating anchor points by modifying the *distance image D*. An anchor point at position c adds a *subtractive function* which is centered at c and is applied to all pixels p of its influence region R:

$$D_p = D_p - f(||p-c||) * k \text{ ; for } p \in R$$

where f is a non-negative decreasing function, $||p-c||$ is the distance between p and c, and k is a scaling factor. The algorithm now determines the *distance image D* in a first phase. The second phase subsequently computes anchor points for visual objects by selecting the maximal distance values on their segments. After selecting each anchor point, the values in the *distance image* around the anchor are modified by the subtractive function.
In Figure 6, the green and red curves denote distances of two distinct visual objects to their segment boundaries. After placing an anchor point for the green object, the subtractive function (in blue dotted line) digs a valley into D and forms a new peaks for the red object (Peak 2 and 3). Finally, Peak 3 is selected as anchor position since it now has the highest quality.

5.2.3 Label Layout

All style-specific algorithms instantiate a generalized algorithm. In the following, we describe only the layout specific realizations of generalized tasks. All layout-specific algorithms strictly prevent label overlaps and intersections of connecting lines. Moreover, layouts can be made more compact. Our extensions to achieve a frame-coherent label layout are presented in the next section.

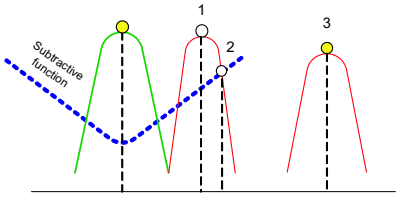


Figure 6: Separation of anchor points.

Generalized Algorithm: For a single frame,

1. Determine the positions of anchor points,
2. Determine the extents of empty space regions,
3. Allocate spatial regions for labels,
4. Compute an initial label layout,
5. Stack labels to eliminate overlap,
6. Resolve line intersections, and
7. Perform layout compaction.

All layout algorithms rely on the positions of anchor points. In Task 2, the extent and locations of the four biggest empty axis-aligned rectangles (left, right, top, bottom) around the graphical model are computed, as the *flush* layouts place labels solely in these regions. The Tasks 3, 4, and 7 are style specific.

Flush-Left-Right Layout: Modifies Tasks 3, 4, and 5.

Allocate spatial regions for labels (Task 3):

(a) Sort anchor points according to the x direction,
(b) Choose a *pivot* point (e.g., mean or median of anchor points), and
(c) Assign labels to the left and right region by comparing their anchors with the pivot point.

Compute an initial label layout (Task 4):

(a) Assign the y position of anchor points to y position of associated labels.
(b) Justify the labels on the bounding box of the graphical model.

Stack labels to eliminate overlap (Task 5):

(a) A recursive algorithm assigns new positions to the labels to eliminate label overlaps and minimize the average vertical length of connecting lines.

For *flush left* and *flush right* layout, we assign to the pivot element a minimal or maximal value. For *flush top* and *flush bottom* layout, the previous algorithm works by exchanging horizontal and vertical directions. Later on, in top and bottom regions, labels are stacked in vertical direction.

Radial Layout: Modifies Tasks 4 and 5.

Compute an initial label layout (Task 4):

(a) Select a center position o (e.g., mean or median of anchor points) and an appropriate radius r for a circle C which encloses the graphical model.
(b) Compute the radial projection of the anchor points on C.
(c) Align the corresponding label on this position. Labels on left-half of C are right-justified and labels on right-half of C are left-justified.

Stack labels up- or downwards to eliminate mutual label overlaps (Task 5).

The radial projection of anchor points produces no intersections of connecting lines. However, labels can overlap or lie very close to each other which is resolved by label stacking. An unbalanced distribution of labels might cause huge label stacks and increase the lengths of connecting lines. The *spring embedding* approach, which is described after the discussion of the individual layout styles, improves the layout considerably.

Ring Layout: Modifies only Task 4.

Compute an initial label layout (Task 4):

(a) Select a center position o (e.g., mean or median of anchor points) and an appropriate radius r for a circle C which encloses the graphical model.
(b) Choose n evenly spaced positions P on the circle.
(c) Determine a bijective mapping from the label set to P which minimizes the distance between labels and their anchor points.

Since the labels are already evenly spaced, there is no need to check for label overlaps for small n and big r.

Silhouette-Based Layout: Modifies Tasks 4 and 5.

Compute an initial label layout (Task 4):

(a) Compute the convex hull of the geometric model and enlarge it (pre-processing step)
(b) Project the anchor points on the edges of the convex hull silhouette boundary S. Choose the closest projection position to the anchor as label position.

Stack labels up- or downwards to eliminate mutual label overlaps (Task 5).

In our application the approximation of the silhouette boundary with convex hulls achieved a better quality compared to other bounding objects (e.g., circles or bounding boxes). But also this approach suffers from uneven label distribution. Again, the spring embedding approach is used to improve this layout.

Spring Embedding Approach

To balance uneven label distribution in *radial* and *silhouette-based* layouts, we use a *force directed* approach [FR91] developed in graph drawing. We define a *repulsive force* between labels aiming to separate the labels, and an *attractive force* that moves the labels close to their anchor points. The configuration is done in a circular fashion. Hence, it is based on *angles* between the labels rather than on *distances*. For two labels v and u, Δ_r refers to an interior angle formed by them with respect to circle center o. The repulsive force f_r is inverse proportional to Δ_r:

$$f_r(\Delta_r) = -k^2/\Delta_r$$

where k is an ideal angle (e.g., 0.2 rad.) between v and u. Moreover, we establish an attractive force f_a between the labels v and associated anchors a_v. Let position p_v be the radial projection of a_v in radial layout. The position p_v on the circular ring attracts the label v. Δ_a refers to the interior angle between v and p_v with respect to o.

$$f_a(\Delta_a) = \Delta_a^2/k$$

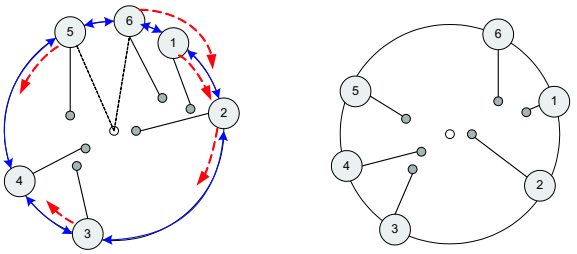


Figure 7: Using *spring embedding* to spread labels in the circle. Before (left) and after configuration (right).

where k is a tolerable angle (e.g., 0.2 rad.) between v and p_v. The algorithm configures the layout in many iterations. The amount of *temperature* t constrains the label displacement. The higher t, the bigger the displacement. The algorithm starts at high t and cools gradually. In each iteration, displacement angles for each label are computed by summing repulsive and attractive forces. After configuration, label overlaps are resolved. In our system, spring embedding approach can be performed in real-time as nearly 30 objects are labeled. We found 20 to 30 iterations enough to achieve an acceptable layout.
Figure 7 illustrates the spring embedding configuration. Filled enumerated circles represent labels which are arranged on a circle, whereas tiny filled circles represent anchor points. Solid blue lines indicate repulsive forces and dashed red lines indicate attractive forces between labels and their ideal positions.

5.2.4 *Line Intersection Elimination*

To resolve intersections of connecting lines, the restriction on *fixed size* labels is a big advantage. For any two intersecting lines, we can interchange their label positions without introducing new label overlaps:

Do until there are no intersections left
  if any two connecting lines intersect
    interchange their label positions

Orthogonal Layout
This style requires axis-aligned connecting lines with bend at orthogonal angles. It can be combined with all *flush* or *circular* layouts:

1. Compute label positions using any layout method,
2. Draw the connecting lines in orthogonal style.
3. Resolve intersections of orthogonal lines, and

The current implementation imposes two restrictions: (i) only one bend is allowed (i.e., connecting lines can employ a vertical and a horizontal segment) and (ii) vertical segments connect anchor points and bends, while horizontal segments connect bends and labels.
In order to detect line intersections in the orthogonal layout, each horizontal segment is tested for intersection with all vertical segments. Every time an intersection is found, the label positions are exchanged. This procedure is continued until no intersections remain.

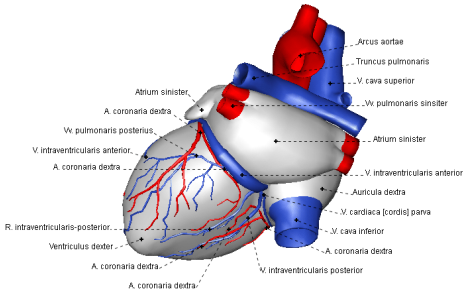


Figure 8: Layout Compaction.

5.2.5 *Layout Compaction*

In order to keep the connecting lines short, this step aims at moving the labels towards their anchor points. We implemented two methods based upon the kind of silhouette they use. The first method approximates the silhouette of graphical model by a convex hull. It computes intersections between connecting lines and the edges of the convex hull and re-targets the labels on the intersection points. Furthermore, label overlaps are resolved analogue to Task 5 in *silhouette-based* layout (see Figure 3-b).
The second compaction method uses the original silhouette boundary, and is preferred only for *flush left-right* layout as the height of labels is much smaller than the width. Each label in the left region is shifted horizontally towards the right until it hits some foreground pixel, or the horizontal distance between the label and the anchor point becomes zero. This test is performed using the *color-code image*. The labels are placed with some margin to the final position. Similarly, the labels in the right region are moved in. New line intersections may arise which are again resolved. After compaction, the labels in both sides closely follow the boundary of the model, and the layout looks more pleasing (see Figure 8).

5.2.6 *Frame Coherent Presentation*

Our discussion so far was restricted to static aspects. In interactive systems the label layout has to be recomputed after the user interacts. An independent layout for individual frames without considering continuity aspects results in *layout flickering*. Jumping labels and anchor points are both irritating and distracting. In order to achieve a *frame coherent* label layout, our algorithms are revised to consider the outcomes from previous frames.

Stabilizing Anchor Points: All of the layout algorithms rely heavily on the positions of anchor points. Therefore, movements of anchor points might induce a global change in the layout. To enhance the frame coherency, a new heuristic for placing anchor points

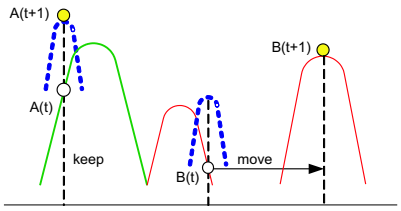


Figure 9: Stabilizing anchor points.

is added: *If possible, keep the anchor points at their previous locations.* However, anchor points must not leave the region of its corresponding visual object and might now reside at a very poor position. In both cases, the anchor points should shift.

In order to implement this new heuristics, we add an *attractive force* which aims at keeping anchor points close to their previous positions. For each anchor point at position c and its associated visual object O_i an *additive function* is applied on the *distance image* D. It affects the distance values for all pixels p of the corresponding object within an influence region R:

$$D_p = D_p + f(||p-c||) * k \quad ; \; p \in O_i$$

where f is a non-negative decreasing function, $||p-c||$ is the distance between p and c; and k is a scaling factor.

This function creates high peaks on previous anchor positions and increases their probability for being selected as new anchor points. In Figure 9, the $A(t)$ and $B(t)$ refer to the anchor points of the green and red object in frame t. In the next frame $t+1$ the additive function (in blue dotted line) modifies D and creates new peaks. The global maxima $A(t)$ and $A(t+1)$ for the green object are identical, so that its anchor point remains stable. However, there is now a new global maximum $B(t+1)$ for the red object, so that its anchor point moves to another location. This illustrates how we try to retain old positions as long as they are acceptable and jump to better candidates otherwise.

Stabilizing Label Layout: The assignment of labels to spatial regions is a very crucial for the appearance of a layout. In order to prevent frequent label jumps between different regions, the pivot point of flush layouts should also be stabilized. However, if it remains steady for a long time while the user interaction continues, the numbers of labels per region can get very unbalanced. To handle this problem, a pivot element is nailed as long as it provides an acceptable ratio of anchor points in two regions, otherwise it is set to new location.

Label Animation: In order to prevent visual discontinuities, changes of anchor points and label positions are animated. We prefer a *slow-in slow-out* interpolation. To avoid a distraction by floating labels within user interactions, layouts can be *frozen* until a new stable point-of-view is chosen.

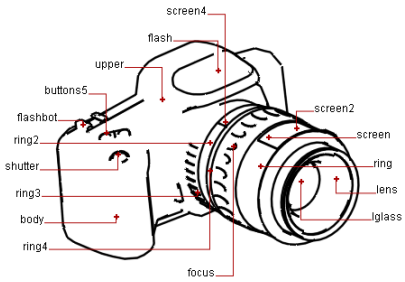


Figure 10: An *orthogonal layout* with NPR rendering.

5.2.7 *Decoration*

By default, we use white as background color and black as text color. For legends, different colors for indices help to find the associated elements. We suggest (i) to use dotted instead of solid lines (otherwise strips between lines show up), (ii) to use line shadows, and (iii) to decrease the color intensity of labels and connecting lines during animations. The system facilitates the user to change the color, size, and style for anchor points and connecting lines.

For creating abstract versions of illustrations, we integrated a non-photorealistic rendering system [HIR+03]. Figure 10 shows the results with orthogonal layout style. Long textual descriptions can be presented in both 3D and in a separate *legend viewer*. Object selection and highlighting is synchronized in both views (see Figure 11).

5.3 Selection of Layout Style

No layout is ideal under all circumstances, however, the knowledge of their specific advantages and constraints helps to select an appropriate one. The choice of a layout for external labeling depends largely on the shape and orientation of visual object, the spatial distribution of anchor points, the amount and distribution of free space available to insert labels, and personal preference for a particular layout.

Determining a suitable layout automatically can be very difficult. Therefore, layout selection is performed

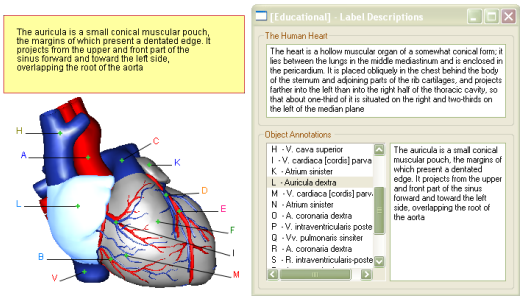


Figure 11: Synchronized legend and 3D viewer.

by the user via real-time previews. It also gives the user more freedom in choosing from a variety of available layouts if one layout does not look promising.

6 EVALUATION

An automatic evaluation of the label layouts can be made on the basis of the following parameters (mainly taken from the field of graph drawing [DBET+99]):

- Number of unlabeled visual objects,
- Number of line intersections,
- Number of label-label overlaps,
- Number of line-label overlap,
- Average length of connecting lines,
- Number of bends in connecting lines,
- Average number of labels which change positions between frames,
- Average label displacement between frames,
- Illustration size and aspect ratio (1 is the best), and
- Frame rate.

Our system measures the values of evaluation parameters to help us in comparing the layouts at runtime. Moreover, a user evaluation should consider the following parameters:

- Contrastive comparison of different layouts,
- Personal layout score (pleasing, symmetry),
- Distinguish automatically generated layouts from hand-made ones,
- Time taken to match the co-referring labels and visual objects, and
- Error rates in the matching process.

7 DISCUSSION AND FUTURE WORK

The *layout compaction* is currently performed in local regions. Improved versions should consider the global distribution of empty space and reduce the amount of label stacking. In our approach, a single anchor point is used for each object. Long, thin, and branched objects are often marked with multiple anchor points, which are connected with branching connecting lines (see Figure 1-Right). For bigger objects (area features) internal labels should be used. Moreover, the system should support the integration of multiple layout style within an illustration. Finally, common semantic classifications of visual objects should be visualized through *labeling grouping* A full-fledged labeling system has to integrate all these aspect, and should be based on the notion of relevance to select an appropriate label number and content dynamically.

REFERENCES

[AdB88] C. Arcelli and G.S. di Baja. Finding Local Maxima in a Pseudo-Euclidean Distance Transform. *Computer Vision, Graphics, and Image Processing*, 43(3):361–367, 1988.

[BFH01] B. Bell, S. Feiner, and T. Höllerer. View Management for Virtual and Augmented Reality. In *Proc. of Symposium on User Interface Software and Technology*, pages 101–110, 2001.

[CMS95] J. Christensen, J. Marks, and S. Shieber. An Empirical Study of Algorithms for Point-Feature Label Placement. *ACM Transactions on Graphics*, 14(3):203–232, 1995.

[CP86] J.M. Clark and A. Paivio. Dual Coding Theory and Education. *Educational Psychology Review*, 3(3):149–210, 1986.

[DBET+99] G. Di Battista, P. Eades, R. Tamassia, , and I.G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, Upper Saddle River, NJ, 1999.

[FP99] J.-D. Fekete and C. Plaisant. Excentric Labeling: Dynamic Neighborhood Labeling for Data Visualization. In *Proc. of SIGCHI*, pages 512–519, 1999.

[FR91] T.M.J. Fruchterman and E.M. Reingold. Graph Drawing by Force-Directed Placement. *Software-Practice and Experience*, 21(11):1129–1164, 1991.

[HIR+03] N. Halper, T. Isenberg, F. Ritter, B. Freudenberg, O. Meruvia, S. Schlechtweg, and Th. Strothotte. OpenNPAR: A System for Developing, Programming, and Designing Non-Photorealistic Animation and Rendering. In *Proc. of Pacific Graphics*, pages 424–428, 2003.

[Imh75] E. Imhof. Positioning Names on Maps. *The American Cartographer*, 2(2):128–144, 1975.

[KT98] K.G. Kakoulis and I.G. Tollis. A Unified Approach to Labeling Graphical Features. In *Proc. of the 14th Annual Symposium on Computational Geometry*, pages 347–356, 1998.

[MS91] J. Marks and S. Shieber. The Computational Complexity of Cartographic Label Placement. Technical Report TR-05-91, Center for Research in Computing Technology, Harvard University, 1991.

[PRS97] B. Preim, A. Raab, and Th. Strothotte. Coherent Zooming of Illustrations with 3D-Graphics and Text. In *Proc. of Graphics Interface*, pages 105–113, 1997.

[Rog92] A.W. Rogers. *Textbook of Anatomy*. Churchill Livingstone, Edinburgh, 1992.

[RP68] A. Rosenfeld and J. Pfaltz. Distance Functions in Digital Pictures. *Pattern Recognition*, 1(1):33–61, 1968.

[RSHS03] F. Ritter, H. Sonnet, K. Hartmann, and Th. Strothotte. Illustrative Shadows: Integrating 3D and 2D Information Displays. In *Proc. of Int. Conf. on Intelligent User Interfaces*, pages 166–173, 2003.

[SPP97] J. Sobotta, R. Putz, and R. Pabst, editors. *Sobotta: Atlas of Human Anatomy. Volume 2: Thorax, Abdomen, Pelvis, Lower Limb*. Williams & Wilkins, Baltimure, 12. English edition, 1997.