Efficient Distance Field Computation using Cluster Trees

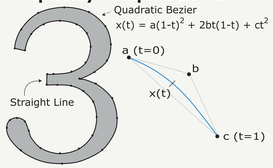
Elena Jakubiak, Tufts University Department of Computer Science • Sarah Frisken, Tufts University Department of Computer Science • Ronald Perry, Mitsubishi Electric Research Laboratories

The Abstract

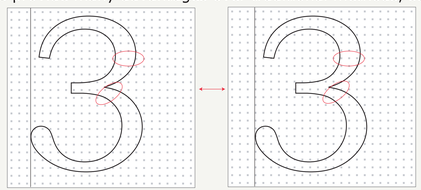
Hierarchical data structures provide a means for organizing data for efficient processing. Most spatial data structures are optimized for performing queries, such as intersection and containment testing, on large, often dynamic, data sets. Set up time and complexity of these structures can limit their value for small, static data sets. We have developed a spatial data structure and associated query algorithm, dubbed Cluster Trees, for providing efficient minimum distance queries from an arbitrary point in space to a small set of Bezier curves. Cluster Trees have been used to significantly reduce the time required to generate an Adaptively Sampled Distance Field (ADF) representing a character glyph such as the letter 'A' from a standard outline representation consisting of lines and Bezier curves.

The Background

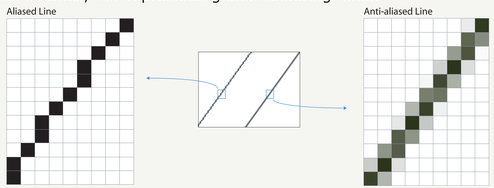
Adaptively Sampled Distance Fields provide many advantages over traditional methods, such as outlines, for representing and rendering fonts.



Typical outline-based fonts are composed of line segments and Bezier curves.



Outlines are used to determine which pixels to color. Outlines may align differently relative to pixel boundaries based upon placement on the page. **Hinting** minimizes these differences.

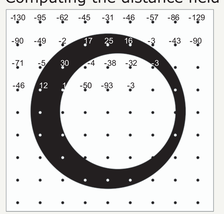


Aliasing can occur whenever a continuous function is sampled at discrete intervals. In images, it can introduce artifacts such as the jagged edges of a line. **Anti-aliasing** can reduce these artifacts, e.g., by 'smoothing' out the jagged edges.

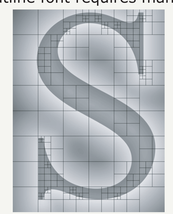
Outline fonts require complicated hinting & suffer from poor anti-aliasing. This leads to excessively blurry and inconsistently rendered glyphs which makes text difficult to read, especially at smaller sizes. Using Adaptively Sampled Distance Fields to render fonts requires no hinting and results in better antialiasing.

The Motivation

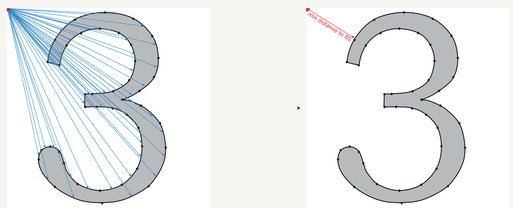
Computing the distance field of an outline font requires many costly queries.



A **Distance Field** is a scalar field representing a shape S with boundary δS that specifies the distance to δS from any point in the field.



Adaptively Sampled Distance Fields (ADFs) provide an efficient means for representing and processing distance fields.



Converting conventional outline-based fonts to ADFs requires computing the minimum distance from each point in the ADF to the glyph's outline. A naïve approach computes the distance from each point to each line and Bezier curve of the glyph's outline and selects the minimum distance. However, this approach is computationally expensive.

Computing the distance field for a character glyph from its outline representation requires many distance queries from the relatively small number of static lines and Bezier curves in the outline. Existing spatial indexing methods are not well-suited to performing minium distance queries on such data.

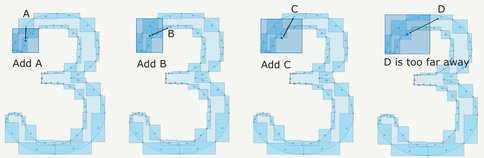
A Solution

Eliminate unnecessary distance computations using a spatial hierarchy of the axis-aligned bounding boxes of the glyphs's Bezier curves.

Building a Cluster Tree

Organize the axis-aligned bounding boxes into a hierarchical spatial data structure.

The axis-aligned bounding boxes of the glyph's Bezier curves form an initial set of leaf nodes of the Cluster Tree. Proximity is used to cluster these leaf nodes into groups.

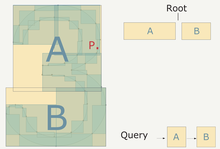


Leaf nodes whose bounding box centers are closer than a maximum pairing distance are clustered into groups. The bounding boxes of these groups form intermediate nodes which are recursively clustered to form the Cluster Tree. The maximum pairing distance is increased at each recursion level.

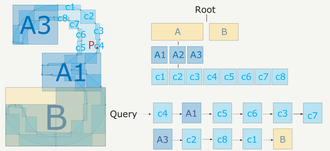
In this example, the bounding boxes of the glyph's segments form the leaf nodes of the Cluster Tree. During the first level of recursion, the leaf nodes are clustered into five intermediate nodes. During the second level of recursion, these five nodes are clustered into two new intermediate nodes. These two nodes become the children of the root node of the Cluster Tree.

Querying a Cluster Tree

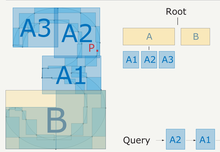
Compute the minimum distance from a given point P to the glyph's outline.



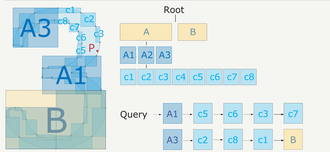
Starting with the children of the root node, create a sorted list of nodes to query. The list is sorted by the minimum distance from each bounding box to P. Here, dist(P, A) < dist(P, B) so A is queried before B.



Repeat this process until a leaf node is encountered. Here, A2 is removed from the sorted list and its children are inserted. P is now closest to c4.



Remove the head of the list and insert its children into the sorted list. Here, A is removed from the sorted list and its three children are inserted.



Update the minimum distance to the glyph's outline when a leaf node is encountered. Stop querying the Cluster Tree when the minimum distance is less than the distance from P to the node at the head of the list. Here, c4 is a leaf node containing a Bezier curve segment δ_{c4}. The minimum distance is updated to dist(P, δ_{c4}). Since dist(P, δ_{c4}) < dist(P, A1), the query is complete.

The Conclusion

Cluster Trees were used to generate ADFs from several standard outline-based font representations. On average, querying the cluster tree required 24 distance-to-bounding box computations and less than three distance-to-quadratic Bezier curve computations per sample point. Computing the distance from a point to a quadratic Bezier curve takes approximately 20 times as long as computing the distance from a point to an axis-aligned bounding box. Hence, for an average glyph containing 40 Bezier curve segments, using Cluster Trees resulted in a 10X reduction in computation during querying. Total ADF generation times, which included building a Cluster Tree for each glyph, were reduced on average by 28 to 49%.