Butterfly Plots for Visual Analysis of Large Point Cloud Data

Tobias Schreck
Technische Universität Darmstadt
tobias.schreck@gris.informatik.tu-darmstadt.de

Michael Schüßler
Siemens ElectroCom
Postautomation GmbH, Berlin
michael.schuessler@siemens.com

Katja Worm
Siemens ElectroCom
Postautomation GmbH, Berlin
katja.worm@siemens.com

Frank Zeilfelder
Technische Universität
Darmstadt
frank.zeilfelder@gris.informatik.tu-darmstadt.de

ABSTRACT

Visualization of 2D point clouds is one of the most basic yet one of the most important problems in many visual data analysis tasks. Point clouds arise in many contexts including scatter plot analysis, or the visualization of high-dimensional or geo-spatial data. Typical analysis tasks in point cloud data include assessing the overall structure and distribution of the data, assessing spatial relationships between data elements, and identification of clusters and outliers. Standard point-based visualization methods do not scale well with respect to the data set size. Specifically, as the number of data points and data classes increases, the display quickly gets crowded, making it difficult to effectively analyze the point clouds.
We propose to abstract large sets of point clouds to compact shapes, facilitating the scalability of point cloud visualization with respect to data set size. We introduce a novel algorithm for constructing compact shapes that enclose all members of a given point cloud, providing good perceptional properties and supporting visual analysis of large data sets of many overlapping point clouds. We apply the algorithm in two different applications, demonstrating the effectiveness of the technique for large point cloud data. We also present an evaluation of key shape metrics, showing the efficiency of the solution as compared to standard approaches.

Keywords: Visual Analytics, Point Clouds, Visual Aggregation, Shape Construction, Shape Refinement.

1 INTRODUCTION

Visualization of 2D point clouds is one of the most basic yet one of the most important task in many data analysis scenarios. Point clouds are an ubiquitous type of data arising in many data analysis tasks. E.g., point clouds may be obtained by plotting pairs of selected attributes of a multivariate data set against each other, obtaining scatter plots which are useful for analysis of correlations, clusters, and outliers. As another example, high-dimensional data sets can be visually inspected by obtaining projections to low-dimensional display space, e.g., by using Principal Components Analysis or Multidimensional Scaling techniques. As a third example, the visualization of geo-spatial data often has to deal with sets of points representing certain locations. Typical analysis tasks in point cloud data include assessing the overall structure and distribution of the data, assessing spatial relationships between data elements, and identification of clusters and outliers.

Standard point-based visualization methods do not scale well with respect to the data set size. More specifically, as the number of data points and data classes increases, the display usually gets crowded quickly. Then, it is very difficult for the user to distinguish different point clouds from each other, or to correctly perceive their shape. Both effects harm the effective visual analysis in standard point cloud visualization approaches.

A typical example where the user is confronted with large sets of point clouds is projection-based visual analysis of high-dimensional data, e.g., in a database exploration application. Assume a database consisting of thousands of multidimensional records in many different groups. Projection of the records to 2D display space, which is desirable for visual analysis of database content, will most likely lead to cluttered and crowded displays of many overlapping point clouds.

The scalability problem may be addressed by aggregation or reduction on either the data level, the visualization level, or both. In this paper, we propose to abstract point clouds by compact shapes, forming so-called Butterfly plots. We demonstrate the suitability of these shapes for the effective visualization of large point cloud data. We develop an algorithm for constructing shapes that (a) enclose all members of a point cloud, that (b) are compact in terms of covered space, and that (c) provide good perceptional properties. We apply the algorithm on two large data sets, demonstrating the effectiveness of the approach. A systematic evaluation of the shape construction algorithm is given, including practical parameter setting recommendations.

2 RELATED WORK

The need to visualize point cloud data arises in many important application areas. Point-based data is obtained e.g., by projecting high-dimensional input data to display space for visual analysis. Principal Compo-

nents Analysis (PCA) [10] or Multidimensional Scaling (MDS) are two popular techniques for projecting high-dimensional input data to low-dimensional display space. Projection-based applications include e.g., the analysis of high-dimensional financial data [3], or multimedia database exploration [13]. Other applications often relying on point cloud data visualization include the analysis of geo-spatial and bivariate data. Multivariate data can be visualized by so-called scatter plot matrices.

Many of these applications require the appropriate visualization of point data. The Bag Plot [14] is a statistical technique that visualizes core and peripheral areas of 2D point data. The over-plotting problem which may occur in large geo-spatial data sets is addressed in [12, 7] by introducing certain geometric distortions and rearrangement of points within their neighborhood. In [8], the authors consider the efficiency of rendering very large sets of point data using improved data structures and rendering algorithms.

A promising approach to visualize large point cloud data is based on visual abstraction of point clouds by shape. The reconstruction of surface from unorganized 3D point sets is a challenge currently addressed in point-based Computer Graphics. E.g., in Computer Aided Design, the reconstruction of shape from CAD models given in point-based representation is addressed [2, 11]. In Image Processing [5], an important problem is the segmentation of shapes in digital images, which is addressed e.g., by means of morphological operations.

In [15], we proposed to abstract large point-cloud data by forming convex hulls over *thinned* point clouds. However, in many cases thinning (data reduction) is not an option, as the whole data set needs to be analyzed and outliers may play an important role. In this paper, we therefore develop an effective point cloud visualization not requiring data reduction.

3 SHAPE ABSTRACTION

In this section, we discuss basic requirements for abstracting point clouds by shapes, and introduce an algorithm for construction of compact shapes completely covering clouds of points.

3.1 Requirements analysis

Traditionally, point clouds are visualized by representing each data item by a symbol. Symbol position represents attribute values, while the symbol itself encodes the class a point belongs to. Usually, form and color is used to indicate class labels. This, however, is not expected to scale with the number of classes. Regarding color, there is evidence that human perception is limited to discrimination only about ten colors simultaneously [16]. The situation is less clear regarding usage of symbol shape, but it can be expected that also, perceptional limits exist. Figure 3 (bottom-left) gives an example

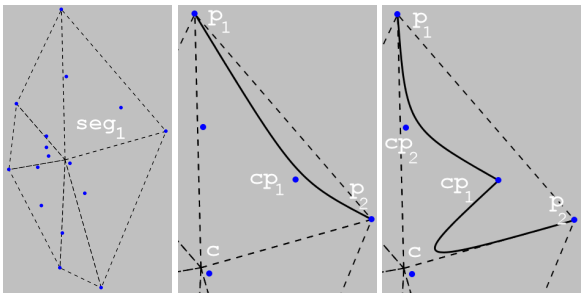

Figure 1: Left: Convex hull enclosing a set of points. The middle and right images show the initial and the first sub-refinement steps respectively, applied on the hull boundary line $\mathbf{p_1p_2}$.

symbol plot of a medium size data set. The number of data items and different classes (45) makes it extremely difficult to perceive even the most essential characteristics of the data distribution.

Abstraction of point clouds by suitable *shapes* is a promising approach to improve a point cloud display in case of many different clouds overlapping each other. An appropriate shape abstraction replaces sets of points by a number of shapes, reducing plot complexity and at the same time, visualizing the essence of the data characteristics.

Shape-based abstraction of large sets of point clouds should provide that the shapes (a) are as *compact* as possible, and (b) are as *predictable* as possible. When visualizing many shapes simultaneously, more compact shapes usually reduce the degree of overlap among the different shapes, thereby supporting shape discrimination. On the other hand, overlap often cannot be avoided on large data sets, even for the most compact shapes. Then, it should be possible for the user to mentally reconstruct as much of the occluded shapes as possible based on the clues given by the non-occluded parts of the shapes. We call this property shape predictability. A tradeoff exists between the two criteria: More compact shapes typically involve boundaries of higher complexity, thereby harming predictability.

Many ways exists to abstract point clouds to shapes. Shapes can be formed to cover the point clouds completely, or just partially. The shapes can be as simple as a bounding disc or rectangle, or be of higher complexity. The shapes may be allowed to contain holes, or not. In [15], we previously proposed to form convex hulls over point clouds thinned for outlier points at the periphery of the point clouds. The convex hulls were found to be more effective that bounding discs and boxes for visual analysis in large point cloud data.

We here are concerned with forming shapes for *complete* coverage of the point clouds. This is desirable as in many applications, it is interesting to analyze the characteristics of outlier points with respect to the whole point distribution. Without outlier removal, the

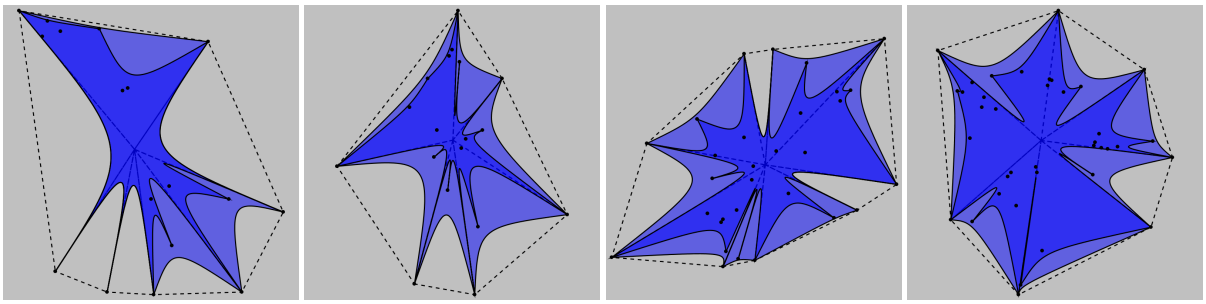

Figure 2: Butterfly plots resulting after the initial (outer shapes) and first sub-refinement steps (inner shapes) have taken place on the convex hulls of several point clouds. Significant reduction of the convex hull area between 56% and 20%, and between 74% and 46% are observed as a result of the first and second refinement steps.

compactness of the convex hull abstraction suffers, as its area is sensitive with respect to outliers. In the next section, we develop a shape abstraction covering *all* points of a given point cloud, which is both compact and offers good perceptional properties.

3.2 Butterfly plot construction

The convex hull [6] is the most compact convex enclosure of a number of points. This makes it a good starting point for a refinement process aimed at reducing the shape area. Our basic idea is to iteratively replace the straight convex hull boundary lines by curves more tightly fitting to the interior points of the point cloud.

We describe our algorithm by means of a constructive example. Consider a set of points $P \in \Re^2, |P| > 2$ with associated convex hull consisting of n boundary lines. Partition the convex hull into n triangular segments by connecting the center of gravity $\mathbf{c}$ of the point cloud with the end points of each convex hull boundary line. Figure 1 (left) illustrates such a partitioning, and Figure 1 (middle) closes up on segment $seg_1 = \triangle(\mathbf{c}, \mathbf{p_1}, \mathbf{p_2})$. Refinement of the convex hull takes place by two steps: Mandatory *initial refinement* of each convex hull boundary line by exactly one curve segment, followed by optional *recursive sub-refinement* of the output of the initial refinement. The optional sub-refinement is controlled by an area reduction-based threshold test, and a global limit of the maximum allowed recursion depth.

The initial refinement stage works by replacing each convex hull boundary line by a curve connecting the line end points and fitting to a control point $\mathbf{cp}$. This control point is found as the point closest to the line to be refined and located inside the respective convex hull segment. In case the given segment does not contain any data points, we take the center of gravity $\mathbf{c}$ as the respective control point. Figure 1 (middle) illustrates the initial refinement of boundary line $\mathbf{p_1 p_2}$ by a curve from $\mathbf{p_1}$ to $\mathbf{p_2}$ controlled by $\mathbf{cp_1}$, which is the data point closest to line $\mathbf{p_1 p_2}$ and contained in seg_1.

After the initial refinement step has taken place, we optionally continue to refine the found curve segment. To this end, we partition each convex hull segment into two triangular sub-segments formed each by the center of gravity, the segment's initial control point, and either one of the segment line end points. Refinement may continue recursively on the sub-segments until either all data points are exhausted, or a termination criterion is met. Figure 1 (right) illustrates the first sub-refinement step applied on seg_1. The respective sub-segments are given by triangles $seg_{1,1} = \triangle(\mathbf{c}, \mathbf{p_1}, \mathbf{cp_1})$ and $seg_{1,2} = \triangle(\mathbf{c}, \mathbf{cp_1}, \mathbf{p_2})$. We refine $seg_{1,1}$ by a curve between $\mathbf{p_1}$ and $\mathbf{cp_1}$ controlled by $\mathbf{cp_2}$, and $seg_{1,2}$ by a curve between $\mathbf{cp_1}$ and $\mathbf{p_2}$ controlled by $\mathbf{c}$.

The refinement of convex hulls by this scheme is expected to show two effects: A recovery of area from the initial convex hull, and an increase in boundary circumference and complexity. We reflect this trade-off by evaluating a test prior to executing each candidate sub-refinement step. Specifically, we execute a candidate sub-refinement step only if

(a) its recursion depth is within a limit ρ, and

(b) it recovers at least a fraction τ of convex hull area.

ρ and τ are useful for balancing area reduction and shape complexity. Note that we evaluate this test only on recursive sub-refinement steps, and not on the initial refinement step which is executed mandatorily. The rationale is to avoid shapes mixing unrefined segment lines with curves, a combination which according to our experiments showed undesirable perceptional effects. Also, the *type of curve* used for refinement needs to be specified. We experimented with different curve types, and found the Bézier Cubic Spline [4], obtained by doubling the control point, giving good results. This curve tightly fits its control point, and runs strictly within the triangle spanned by the given start, end, and control point. The latter property in conjunction with our convex hull segmentation scheme provides that the obtained shape completely encloses the given point cloud.

Algorithm 1 gives our convex hull refinement algorithm. It consists of the main procedure `butterfly` taking as input a point cloud P, an area reduction thresh-

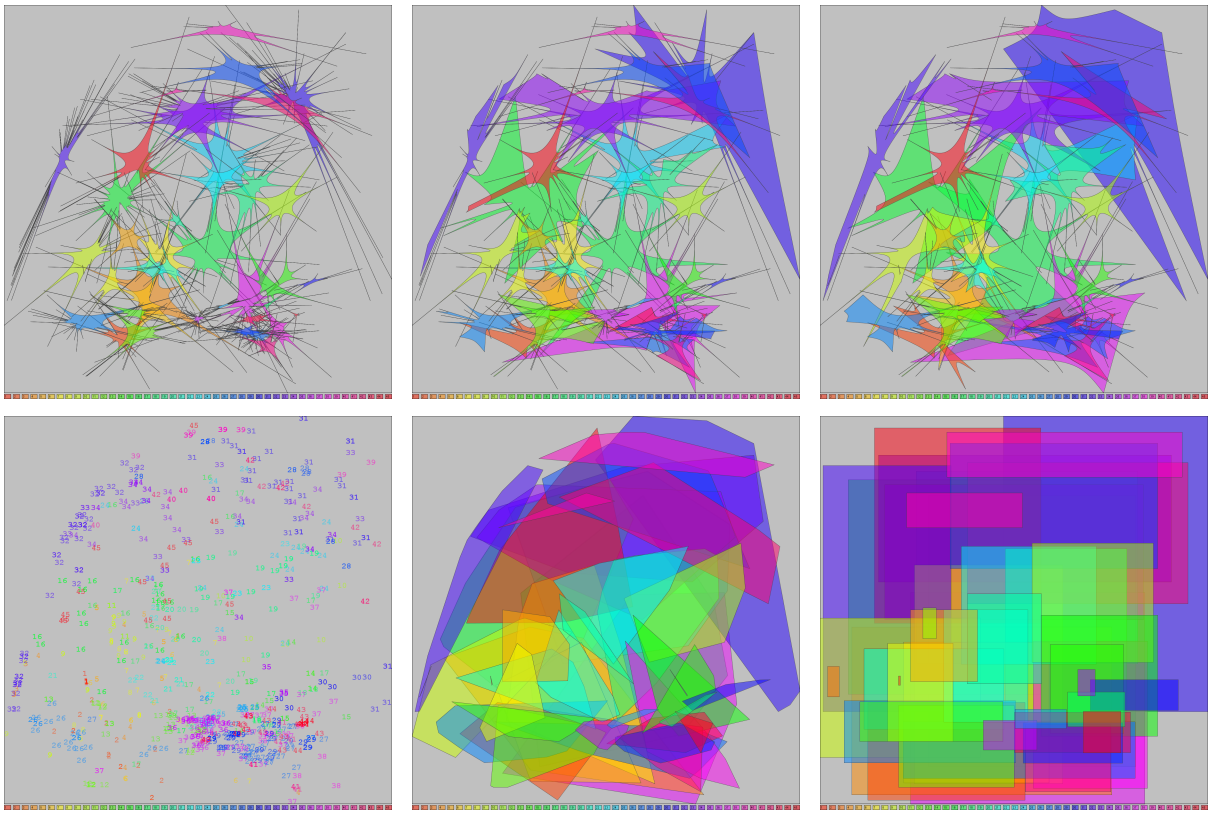

Figure 3: Butterfly plots obtained using refinement thresholds $\tau = \{0.00,\ 0.02,\ 0.04\}$ and unlimited recursion depth, for a data set consisting of 45 classes (top row, left to right). The bottom row shows plots using labeled point clouds, and enclosing convex hulls and minimum bounding boxes. The Butterfly plot approach improves over these standard point cloud visualization methods, allowing effective visual point cloud analysis.

old τ, and a recursion depth limit ρ. Note that in procedure `refine` the term $|l \cup curve(l.p1, cp, l.p2)|$ denotes the area enclosed between a (segment) line l and a refinement curve connecting l's end points. It corresponds to the area reduction achieved by applying a given refinement step. Figure 2 shows exemplary results obtained for several point clouds from a data set to be further discussed in Section 4.2. The shapes were obtained by setting $\tau = 0$ and $\rho = \{0, 1\}$, respectively. We observe the shapes are compact and possess a smooth, predictable boundary of limited complexity. We decided to call the obtained shapes Butterfly plots, as one can recognize certain similarities between the shapes and the insect, on an abstract level.

4 APPLICATION

We next present two Butterfly plot applications, demonstrating the effectiveness of the technique.

4.1 Database exploration data set

The first application is in *database exploration*. Visual database exploration methods often provide graphical representations of the database objects and their interrelationships. In [13], the visual exploration of a database of more than 850 3D CAD models [9] was demonstrated by means of PCA projection of corresponding object feature vectors. We visualize that database to support exploration of the relationship between different object classes under a given feature vector representation. We generate Butterfly plots of the database as follows. We first map the database to 2D by PCA analysis of the database feature vectors (specifically, *complex-SH* 3D features [1] were used). Together with a classification associated to the database [9], 45 2D point clouds are obtained, for which we generate a Butterfly plot each. We sort the resulting shapes decreasingly by size, and give each a distinct color sampled equally from the rainbow palette. An overall Butterfly plot for the full database is obtained by rendering all individual Butterfly plots in order, using the associated color at medium transparency. Note that color here supports visual class discrimination but does not carry additional information.

The top row in Figure 3 shows Butterfly plots obtained for $\rho = \infty$ and $\tau = \{0.00,\ 0.02,\ 0.04\}$. The leftmost plot represents the maximum refinement possible for a Butterfly plot, yielding a skeleton-like abstraction of the point clouds allowing effective visual perception of two salient data characteristics: Location of point

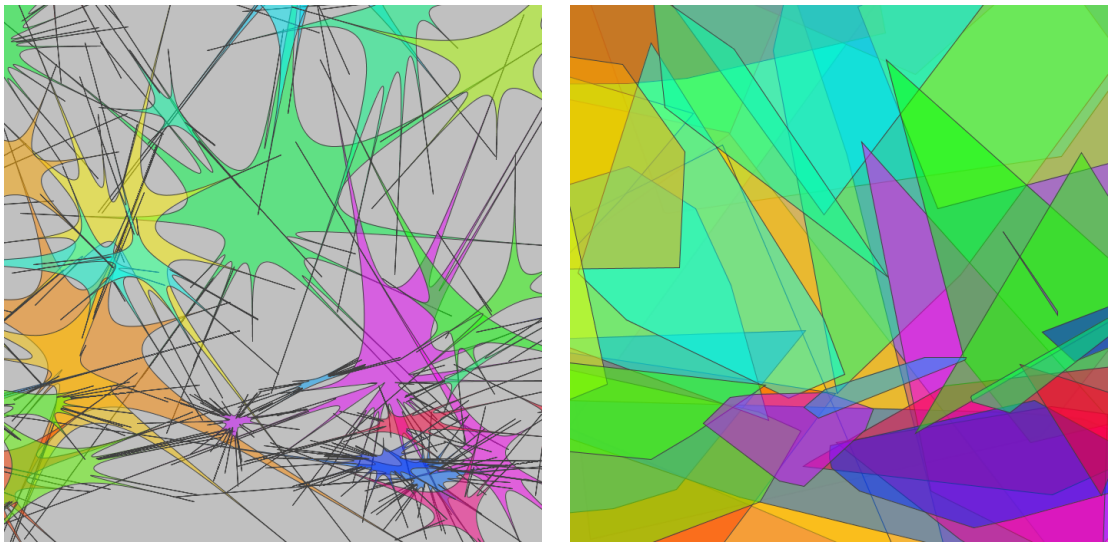

Figure 4: Zooming into a dense area of the Butterfly ($\tau = 0$) and convex hull plots from Figure 3.

cloud centers, and distribution of cloud member points. The center is easily perceivable by the shapes' main area, which is also pointed to by the various curve segments. By construction, we know that each corner point of the shape contains at least one data point. Thereby, we can follow the elongated attachments to analyze the relation between point cloud center and member points. Butterfly plots for $\tau = 2\%$ and $\tau = 4\%$ are shown in Figure 3 (middle and right image in top row). As expected, less refinement takes place, resulting in less compact shapes which emphasize to a larger extent the degree of outlier distribution: Usually, for more scattered point clouds, Butterfly plots of larger area result. The user can be allowed to adjust parameters ρ and τ interactively to analyze both data characteristics.

For comparison, Figure 3 shows plots of the data using colored labels (bottom left), and enclosing convex hulls (bottom middle) and minimum bounding rectangles (bottom right). Neither of these plots is effective for analyzing the given data set. In the colored label plot, it is extremely difficult to discriminate the 45 different class labels or to asses the class distributions. In the other two plots, due to the size of the shapes, too much overlap prevents effective usage (cf. also 4 for a closeup.) The Butterfly plots, on the other hand, manage to effectively visualize the most important data characteristics: The number of classes, the distribution of class member points, and the relationship between the classes can easily be analyzed.

4.2 Pattern recognition data set

The second application is in the field of *pattern recognition*. The task under consideration is recognition and identification of postal stamps in images of mail pieces. Automatic stamp recognition is needed in postal sorting machines, e.g., for checking the correct franking value. The system described here is part of automatic mail sorting systems operating in several countries.

Part of the stamp recognition process is extraction of color features from possible stamp regions in the mail piece image, and matching these features to a set of known prototypes from a stamp image database. The color value of each pixel in a region is entered into a color histogram, then relative frequencies in each histogram bin are used as features. Therefore, the feature vectors have the same size as the histogram, which is typically a few hundred. For purposes of stamp image analysis and recognition performance tuning it is desirable to visualize these feature vectors with their respective class memberships. Besides the high dimensionality of the respective feature space, the large number of stamp classes poses additional visualization problems. Typically, the stamp image database contains some 100 to 500 stamp types (classes), each represented by up to 10 specimen.

The feature vectors of the samples of a given class should, in theory, be all identical. In practice, however, variations in illumination, image background and stamp region detection lead to significant deviations within the sample feature vectors. During performance tuning of the stamp recognition system, often the question arises why some sample images of a given class could not be recognized correctly but were rejected by the classifier. An analysis of such cases is often tedious and involves investigating feature and classifier properties.

To support this process, we designed a Butterfly plot-based visualization for analysis of the deviations between class prototypes and rejected samples. Figure 5 (top) displays the PCA-projected prototypes of 109 stamp classes in a Butterfly plot. Four different stamp classes of which samples were rejected by the classifier are highlighted by increased Butterfly color opacity. The rejected samples are marked by their true class la-

Input: 2D point set P, refinement threshold τ, recursion limit ρ
Output: Closed sequence of curve segments S

```
procedure butterfly(P, τ, ρ):
  S ← {}
  point c ← c(P) /* find center of gravity */
  polygon CH ← CH(P) /* find convex hull */
  /* calculate minimum area reduction required */
  area a ← τ * | CH |
  /* loop convex hull boundary lines and refine */
  for each line l ∈ CH boundary do
    S ← S + refine(P,l,c,0,ρ,a)
  return S
```

Input: Point set P, line l, point c, recursion level r, recursion limit ρ, area a
Output: Set of connected curve segments S

```
procedure refine(P, l, c, r, ρ, a):
  control point cp ← getControlPoint(P,l,c)
  /* evaluate if area reduction threshold is met */
  boolean area_accept ← (|l ∪ curve(l.p1,cp,l.p2)| ≥ a)
  if ( r > 0 ∧ (¬area_accept ∨ r > ρ) ) then
    /* discard refinement step: area reduction of non-initial refinement step below threshold, or recursion limit exceeded */
    return {}
  if (cp = c) then
    return curve(l.p1,c,l.p2) /* P exhausted for this segment */
  /* recurse */
  S1 ← refine(P,(l.p1,cp),c,r+1,ρ,a)
  S2 ← refine(P,(cp,l.p2),c,r+1,ρ,a)
  if ( S1 = {} || S2 = {} ) then
    /* at least one sub-refinement failed */
    return curve(l.p1,cp,l.p2)
  return S1 + S2 /* sub-refinement successful */
```

Input: Set of points P, line l, point c
Output: Control point cp
$d()$: Distance between a point and a line

```
procedure getControlPoint(P, l, c)
  point set P_cand ← P ∩ △(l.p1,l.p2,c)
  /* remove line end points, keep center of gravity */
  P_cand ← P_cand \ (l.p1,l.p2)
  /* find nearest neighbor to l from P_cand */
  cp ← p ∈ P_cand | ∄q : q ∈ P_cand ∧ d(q,l) < d(p,l)
  return cp
```

Algorithm 1: Butterfly plot construction.

bels, and connected to their respective class centers using straight lines. The display allows the visual analysis of the problematic samples in context of the whole classifier data.

Interestingly, in this example data set we recognize that many of the displayed rejected samples have similar deviations from their class center. A view focusing on the rejected classes is shown in Figure 5 (bottom left). We learn that similar deviation patterns are most dominant in classes 22 and 88, while for classes 6 and 77, more diverse deviation patterns occur. We suspect systematic differences between the stamp prototypes used by the classifier, and the stamps seen on the rejected sample mail pieces. The reasons for these differences may be various, but the fact that there are systematic differences is very helpful and motivates further analysis. We have implemented interaction functionality which enables the user to perform drill-down analysis. E.g., we may choose to further inspect the lower bundle of rejected class 77 samples. To this end, the system can identify and highlight those prototype classes which (in the PCA-projection) interfere most with the mass of rejected samples. This is done for class 77 in Figure 5 (bottom right). Such drill-down analysis of the relationship between rejected samples and interfering neighboring classes in projected feature space is expected to be a useful tool in the process of classifier tuning.

5 EVALUATION AND DISCUSSION

Generation of Butterfly plots requires specification of the area reduction threshold τ and the recursion depth limit ρ. In experimenting with the data discussed in Section 4.1 useful parameters were found quickly, by testing just a small number of settings. Specifically, we found that $\rho \in \{0,1\}$ and $\tau \in [2\%,\ldots,10\%]$ gave useful results in that application. Generally, the choice of parameters will depend both on data and given task. For dense, mixed class distributions, lower area recovery thresholds and larger recursion limits produce leaner shapes, avoiding excessive overlap. For less crowded data sets, one can use higher thresholds.

In experiments we observed important shape metrics at different parameter settings. A series of plots was generated for the CAD database discussed in Section 4.1 by varying τ while allowing unlimited recursion depth. Figure 6 (top) shows the average size and overlap[1] of the Butterfly plots, relative to the corresponding convex hull plot. At $\tau = 0.00$, the Butterfly plot shows just about 7% (30%) of the area (overlap) of the corresponding convex hull plot. Increasing τ to 0.15 drives up relative area (overlap) of the Butterfly plot to 47% (55%). After that, the metrics increase slower, converging to 54% (63%) relative area (relative overlap) at $\tau = 0.30$. After that the metrics remain stable, as practically all candidate sub-refinement steps are rejected and only initial refinement steps are performed. We also observed the average number of curve segments, relative to the average number of convex hull segment lines (Figure 6 (bottom)). This metric can be interpreted as a measure of shape complexity. At $\tau = 0$, the Butterfly plots on average consist of 2.4 times the number of segments of their corresponding convex hull plots. Increasing τ quickly leads to a reduction in Butterfly plot complexity.

Similar results were obtained for the data set from Section 4.2. The analysis of these shape characteristics may serve for automatic determination of good parameter settings. Assume that shape complexity would be

[1] Size is measured by the number of pixels covered by a shape, and overlap is measured as the average number of shapes covering each non-empty pixel. Measures taken on a $1200 * 1200$ pixel display.

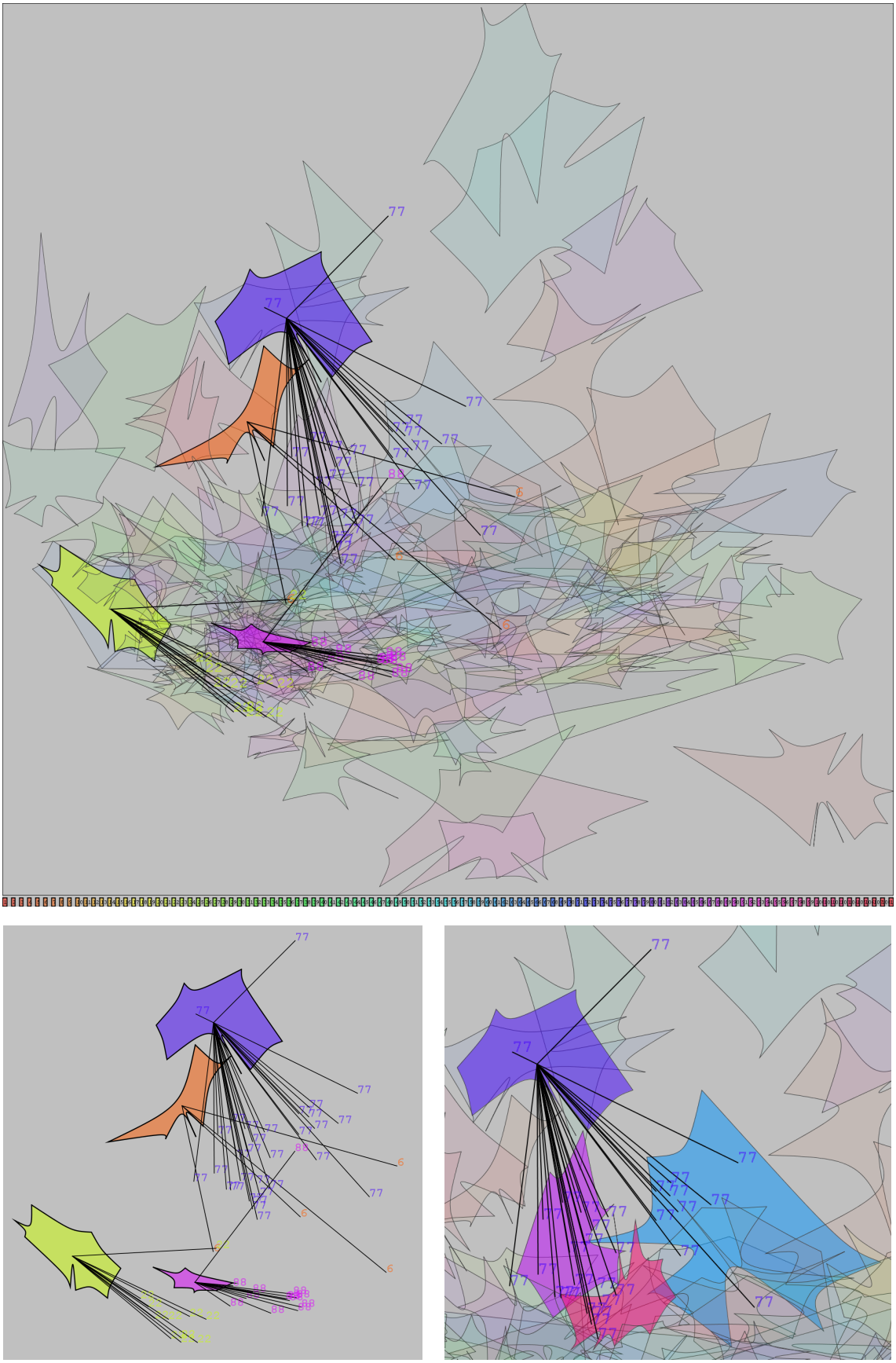

Figure 5: Visualizations based on the Butterfly plot, applied to a pattern recognition application.

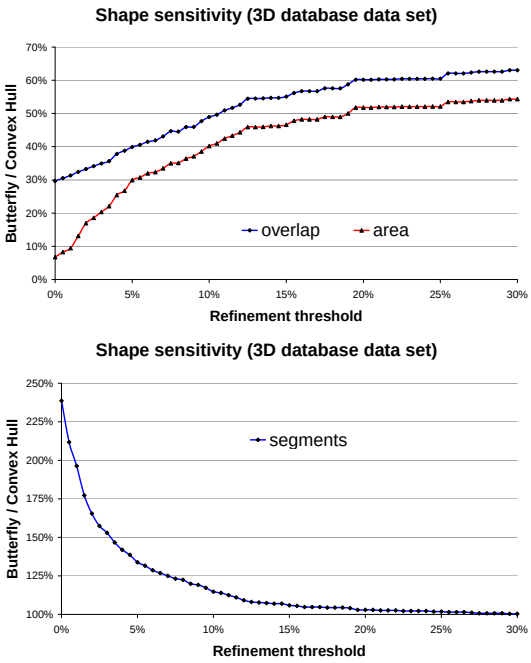

Figure 6: Evaluation of key Butterfly plot metrics.

the most important aspect to control in a given application. Then, by observing the shape complexity behavior, the system could apply the well-known elbow criterion to find $\tau = 0.05$ as a good parameter selection. It is also possible that the user specifies an overlap or shape complexity target which the system in turn uses to determine τ and ρ.

6 CONCLUSION

We addressed the problem of visualizing large point cloud data by abstraction to compact enclosing shapes. We introduced an algorithm that recursively refines the convex hull of a point cloud by curve segments, adapting to the given data. Two parameters allow to adjust the shape construction process to data and user requirements. By application and evaluation, it was shown that the generated shapes are an effective tool for visual analysis of large point cloud data.

One of the greatest challenges in the visualization of large point cloud data lies in optimizing the perception of shape discrimination, in presence of high degrees of overlap between the point clouds. In future work, usage of color for shape discrimination could be optimized, considering the perceptional and spatial proximity of shapes and color. Also, additional shape construction methods can be thought of, and their compactness and perceptional properties should be studied. We currently work on employing appropriate distance field representations for point cloud visualization in conjunction with advantageous spline operators, and already obtained promising first results.

A widely unsolved problem is the visual analysis support for extremely large point data sets. In certain classification applications, tens of thousands of point pairs may arise, each forming a separate class. As such data distributions pose challenges to any point-based visualization technique in general, we presume that in this case suitable data preprocessing is required. Also in this case, computational efficiency concerns may arise, a point which we did not address in this work. Finally, it would be a good idea to do user studies on the perceptional effects of different shapes in different user tasks. Results thereof could guide future work in constructing shapes for visual analysis of large point cloud data.

ACKNOWLEDGMENTS

We thank Tatiana Tekušová of Fraunhofer IGD for valuable suggestions and for discussion. We also thank the anonymous reviewers for their comments which helped to improve this work.

REFERENCES

[1] B. Bustos, D. Keim, D. Saupe, T. Schreck, and D. Vranic. An experimental effectiveness comparison of methods for 3D similarity search. *Int. Journal on Digital Libraries*, 6(1), 2006.

[2] J. Daniels, L. Ha, T. Ochotta, and C. Silva. Robust smooth feature extraction from point clouds. In *Proc. IEEE Shape Modeling International*, 2007.

[3] T. Dwyer and D. Gallagher. Visualising changes in fund manager holdings in two and a half-dimensions. *Information Visualization*, 3(4):227–244, 2004.

[4] Gerald E. Farin. Triangular bernstein-bézier patches. *Computer Aided Geometric Design*, 3(2):83–127, 1986.

[5] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 3rd edition, 2007.

[6] R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[7] A. Herrmann and D. Keim. The gridfit algorithm: An efficient and effective approach to visualizing large amounts of spatial data. In *Proc. IEEE Visualization*, 1998.

[8] M. Hopf and T. Ertl. Hierarchical splatting of scattered data. In *Proc. IEEE Visualization*, 2003.

[9] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani. Developing an engineering shape benchmark for cad models. *Computer-Aided Design*, 38(9):939–953, 2006.

[10] I. Jolliffe. *Principal Components Analysis*. Springer, 3rd edition, 2002.

[11] I. Kyriazis, I. Fudos, and L. Palios. Detecting features from sliced point clouds. In *Int. Conference on Computer Graphics Theory and Applications*, 2007.

[12] C. Panse, M. Sips, D. Keim, and S. North. Visualization of geospatial point sets via global shape transformation and local pixel placement. *IEEE Transactions on Visualization and Computer Graphics*, 12, September-October 2006.

[13] J. Pu, Y. Kalyanaraman, S. Jayanti, K. Ramani, and Z. Pizlo. Navigation and discovery in 3d cad repositories. *IEEE Computer Graphics and Applications*, 27(4):38–47, 2007.

[14] P. Rousseeuw, I. Ruts, and J. Tukey. The bagplot: A bivariate boxplot. *The American Statistician*, 53(4):382–387, 1999.

[15] T. Schreck and C. Panse. A new metaphor for projection-based visual analysis and data exploration. In *IS&T/SPIE Conference on Visualization and Data Analysis*, 2007.

[16] C. Ware. *Information visualization: Perception for Design*. Morgan Kaufmann, 2000.