Line Drawing Simplification by Stroke Translation and Combination

Yun Chien, Wen-Chieh Lin, Tsung-Shian Huang, and Jung-Hong Chuang
Department of Computer Science, National Chiao Tung University, Taiwan

ABSTRACT

In this paper, we propose a new algorithm for simplifying line drawing sketches. First, we segment the strokes at the points of large curvature if desired. Then, we perform a low-pass filter and use the result to assign a weight to every stroke. The strokes are moved to the position of the higher weight. After that, we find the stroke pairs and combine them to reduce the total number of the strokes, resulting in a cleaner line drawing art. This system also cuts down the disordered and confusing small strokes and combines them to form long strokes.

Keywords: Stroke simplification, stroke translation, stroke combination

1. INTRODUCTION

The process of making the digital art usually contains three parts, the sketch, line-art, and coloring; as shown in Figure 1. The sketches are usually used to demonstrate the idea of the whole picture. It can be drawn freely and quickly. Next, the artists must make the picture "cleared". They draw the contour of the art carefully using the beautiful lines. Finally, they color the line-art by using the base color to fill the line-art, and then adding the shadows and details to finish the art.

Figure 1. The process of making the digital art: sketch, line-art, and coloring.

The process of making the sketch to the line-art, however, could be very time consuming because it is nearly impossible to draw a long beautiful stroke directly. To prevent the shiver of hand, the stroke must be drawn in one breath. Artists usually draw many short beautiful strokes and link them carefully, resulting in a complete and pretty single stroke. Drawing the line according to sketches seems easy but it is very time-consuming. This motivates us to study how to translate the sketch into line-art automatically. Although it is intuitive for human to understand the shape from lots of discontinuous and disordered lines, it is very hard for the computer. The goal of our study is to find out a method to simplify the sketch to a fewer strokes' line-art, making the contours to be as long as possible and close to the artists' instinct of linking the short strokes and reducing the density of the hatching.

In this paper, we provide a simple algorithm to simplify the strokes, especially for the sparse strokes. We compute the attributes of the strokes, such as tangent and curvature, then use a low-pass filter to gather the sparse strokes together. Finally, we find out the stroke pairs by the attributes of the strokes and combine the stroke pairs to obtain a simplified line-art.

The contributions of our line simplification can be summarized as follows:

- Present a unified simplification framework for contours and hatching.
- Present a stroke simplification based on a low-pass filtering.

Fifth International Conference on Graphic and Image Processing (ICGIP 2013), edited by Yulin Wang, Xudong Jiang, Ming Yang, David Zhang, Xie Yi, Proc. of SPIE Vol. 9069, 90690X · © 2014 SPIE
CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2050273

2. RELATED WORK

Several novel techniques have been proposed for the simplification of line-drawing. We classify them into three groups: progressively improving method, density-based simplification, and curve extracting method.

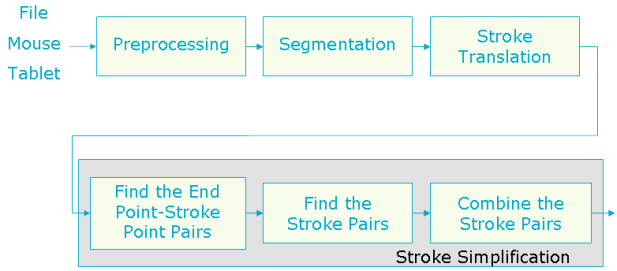


Figure 2. The framework of our system.

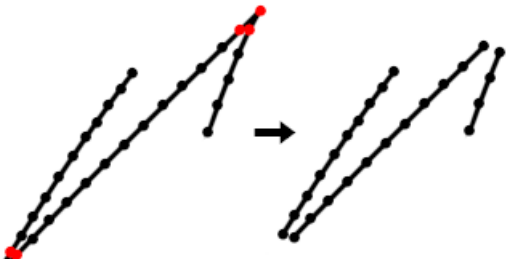


Figure 3. Segmentation.

2.1 Progressively Improving Method

These systems simplify the strokes while users are drawing. New curves that the users just add are used to modify the old curves. Baudel's design is based on patterns extracted from the artists' drawing and editing [2]. It allows input with many strokes to modify a single stroke. Igarashi et al. propose a method to beautify the strokes one after another by considering the geometric constraints among the curves [8]. They also provide multiple candidates to avoid the ambiguity. Kara et al. use conceptual sketches to update the existing 3D template models [9]. They modify the points on the original models by using the attraction of the points from the input curves. A common problem of these methods is that the artists must give up their natural drawing style, which could disrupt their idea of drawing.

2.2 Density-based Simplification

There are several studies that are based on the density control of the line rendering. Preim and Strothotte sort the lines with respect to the screen position, local density and the line length [12]. They cut down the strokes according to this order. Deussen and Strothotte propose a method for drawing the trees [4]. They used the depth information of a tree model to simplify the foliage. Praun et al. created level-of-detail (LOD) for the hatching [11]. They construct a sequence of mipmapped hatch images corresponding to different tones, collectively called a "tonal art map". They use the textures of hatching which fit the user-defined tones. Wilson and Ma created the complexity map of the screen image. They can render the 3D model in different style such as silhouette or hatching [17]. Grabli et al. also create a density map [6]. They use the density map and other information selected from the 3D scene (silhouettes, depth, etc.) to evaluate the significance of the lines and delete the least significant lines. Cole et al. use a "priority buffer" to determine the significant of the lines [3]. These methods only reduce the existing lines and do not add the new lines, which are not suitable for the sketch drawn by the artists, because the artists usually use many short lines to form a long line.

2.3 Curve Extracting Method

Many techniques for drawing simplification have been proposed to extract the geometric curves from completed sketches. Saund extracts the arcs from the curve using "curve element tokens" and then uses the least-squares arc to fit the new curve [15]. Their results, however, are too limited due to the presentation of the arcs. Rosin proposes three aspects to group the strokes: continuation, parallelism and proximity, but they do not provide good experiments [14]. Barla et al. use the concept of the "Ɛ-line" and "Ɛ-group" to simplify the line-drawings [1]. However, the Ɛ-line cannot fold onto itself, and hence is not applicable to the folding or self-intersecting curves. They use the proximity of the curves as the constraint to combine the curves. Pusch et al. divide the curve into small rectangular regions until the region can only be presented by a single curve [13]. Later, they link the small boxes to get new curves. They cannot process the self-intersecting curves, neither. Shesh and Chen use the information of the overlapping, parallelism, and proximity of the strokes to combine the strokes [16]. Their concept of overlapping can be weak because the contour curves may not overlap that much. Orbay and Kara propose a new method not only simplifies the sketches but also beautifies them [10]. They use trainable stroke clustering which uses the neural network that first takes the input of the feature extracted from the stroke and then decides whether the two strokes should be grouped together. Their feature contains three parts: continuation, parallelism and proximity. They focus on the contour of sketches and do not mention the hatching.

3. ALGORITHM

3.1 Overview

We provide a simple algorithm to simplify the strokes. First, we calculate the attributes of the stroke's points, such as tangent and curvature. Then we segment the strokes that are too zigzag or winding, hence we truncate the points with large curvature. Third, we use a low-pass filter, such as Gaussian kernel to move the sparse or overwriting strokes more close such that we can simplify them with a shorter distance tolerance. Then we find out the end point-stroke point pairs and stroke pairs by checking the tangent and position of the strokes. Finally, we combine the stroke pairs and obtain a

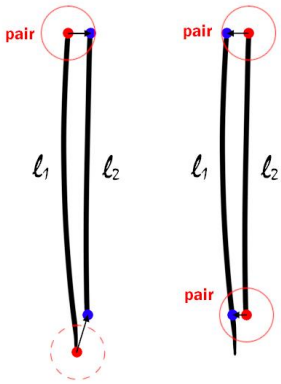


Figure 4. Pairs of end-points for two strokes.

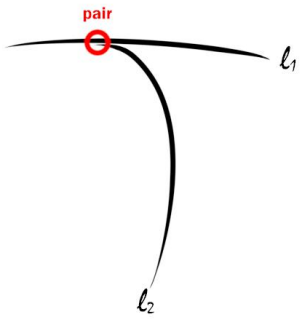


Figure 5. T junction

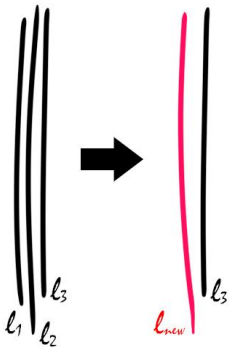


Figure 6. Problem of the order of combination.

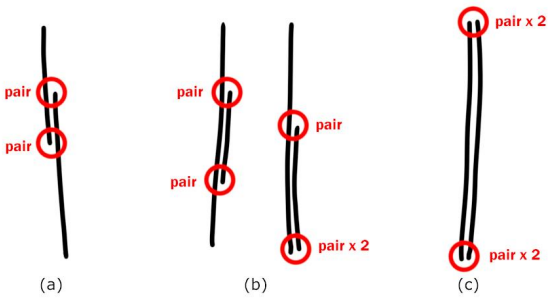


Figure 7. Three cases of combining two strokes

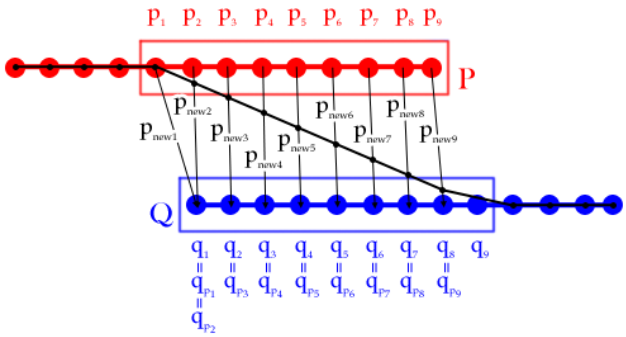


Figure 8. Interpolation of Case A

simplified line drawing art. Figure 2 shows our framework. The input of our system are strokes, and each stroke consists of a serial points with radius. The strokes are rendered using Hertzmann's method [7].

3.2 Preprocessing

If the input points of a stroke are too few or too sparse, we need to do the subdivision. We did not use the curve fitting or regular sampling because we do not have the parametric form of the strokes and we want to make it simple. To do the subdivision, we first calculate the distance between the adjacent points, and insert a new point between them if the distance is larger then a threshold. The newly added point is the midpoint of two adjacent points. The subdivision is recursively applied until all the lengths of segments are smaller than the threshold.

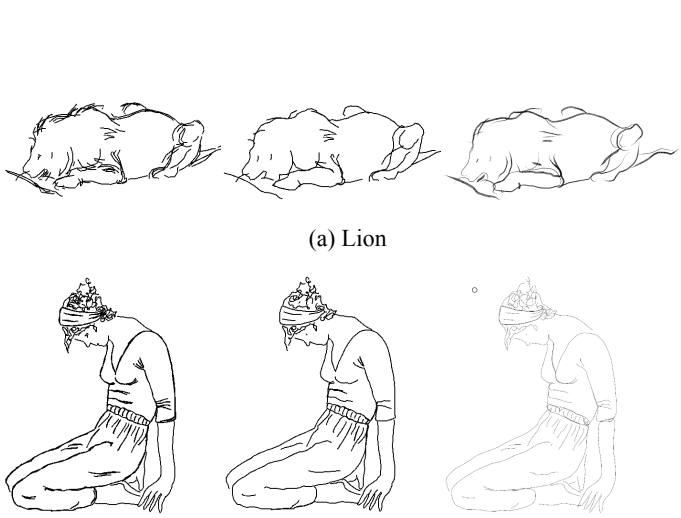
3.3 Segmentation

Because the winding and zigzag stroke could make strokes hard to be combined, this kind of stroke needs to be segmented into several new strokes. We have found out that the curvature becomes large at the bent point of a stroke. Apparently, we need to cut the stroke at the points of high curvature. Unfortunately, if we just separate a stroke at the point of the largest curvature, the end points' tangents cannot present the alignment of the stroke well. It still causes the wrong combination of the strokes, so we cut out all of the parts with large curvatures. We remove the points if its curvature is larger than a user-specified threshold. We did not consider the continuity of the strokes after they are segmented because we think it is not important for the contours or hatching. Figure 3 displays the effect of segmentation.

3.4 Stroke Translation

When artists want to express a thick stroke, they usually draw many strokes at the same place to present the thickness. It is hard to combine the strokes because the strokes are sometimes just too sparse or overwriting. We need to make these strokes closer. Thus, we draw all strokes in a texture to accumulate the weighting, and a low-pass Gaussian filter is applied to this texture. For each stroke, we find the best offset with the highest weighting in a limited nearby position. All the points of the stroke are shifted with the same offset instead of individual offsets since we want to preserve the tangent of the line segments of the strokes.

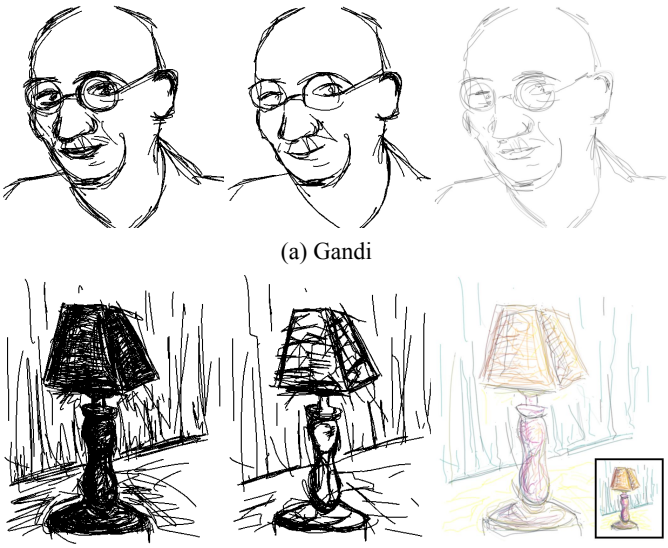
3.5 Stroke Simplification



(a) Lion

(b) Woman

Figure 9. Comparing simplification results of ours and Barlia et al. Left side is the original strokes, the middle is our result, and the right side is the result of Barla et al. [1]



(a) Gandi

(b) Lamp

Figure 10. Comparing results of ours and Shesh. Left side is the original strokes, the middle is our result, and the right side is the result of Shesh and Chen. [16]

The stroke simplification consists of pairing and combination. The stroke pairing is to find the candidates, then the stroke combination uses the best pair to combine and create a new stroke. The system repeats this process until all the strokes are processed.

In stroke pairing, we check the difference between the end points of two strokes. Figure 4 shows that l_1 's end points can find one point pair from l_2, and l_2 's end points can find two point pairs from l_1. Each stroke contains two end points, so there are at most four pairs between two strokes. In Figure 4, there are only three pairs. Then we use two error metric, proximity and parallelism, to check the pair of these end-points. The proximity is the spatial difference between the strokes. It is the distance between end-point and its pair-point. The parallelism is the angular difference between the strokes. It is calculated by the dot product of normalized tangents of an end-point and its corresponding pair-point. The user can set the thresholds of the proximity and parallelism. Then system records the pairs of end-point and its pair-point if they do not exceed the thresholds.

After getting all the candidates of stroke pairs, system checks whether the strokes can be combined together or not. Two strokes can be combined if there are two or more pairs of end-point between them. On the other hand, two strokes with only one pair of end-point cannot be combined. It means that the strokes are arranged as a T junction as shown in Figure 5. They should not be combined into a single stroke.

The order of the combination is also important. After the combination, the strokes may change and cannot be combined with another stroke pair candidate. Figure 6 shows the situation. According to the drawing principles summarized by Fu et al. [5], artists mostly draw from rough to fine. They start with a rough sketch and then gradually refine the drawing strokes, so we prefer to delete the short and minor strokes first. A weighting texture is applied, minor strokes are assigned with small weights. Our system calculates the weighting sums of the points on a stroke and weights it with the length of the stroke. For a stroke pair, the strokes with smaller value are chosen to represent the weight of the stroke pair. And the strokes are sorted in increasing order of the weight. Then the strokes with small weight are combined first.

Our system iteratively chooses two strokes to be combined as a new stroke. There are three cases to combine two strokes. Figure 7 displays these cases. Case A is that the two strokes only have two point pairs and the end-points belong to different strokes as shown in Figure 7(a). Our system creates two sequences of the points for both strokes in the overlapping area. Then the interpolation of two sequences is performed to create new points for combined stroke, as shown in Figure 8. The point positions and radiuses of the new stroke in the overlapping area are all interpolated. After combining two strokes, the system needs to subdivide the stroke and calculate the new tangent for the next stroke combination. The points outside overlapping area do not get modified during this processing. For Case B (Figure 7(b)), the system only chooses the longer stroke. For Case C (Figure 7(c)), it is free to choose any one of the strokes as the new stroke. After combination, the average weight of the stroke is updated, and the strokes are sorted again. Then the first pair in the list is chosen for next combination. This process is repeated until all of the stroke pairs are processed.

4. RESULTS

Here we compare our simplification results to the ones proposed by Barla et al. [1] and Shesh and Chen [16]. Barla et al. used the definition of Ɛ-line to combine the strokes. Shesh and Chen [16] found the point pairs and used the percentage of overlapping to combine the strokes. We do not use the special definition nor the overlapping percentage to justify if the strokes should be combined or not. Instead, we find the end point-stroke point pairs to combine the strokes. Moreover, we use the filter result to centralize the strokes in some overwriting cases, making us easier to combine the strokes. We do not use the attribute of the color, neither. All results are rendered in 512 x 512 resolution.

First, we compare our results with the results of Barla et al. [1]. In Figure 9(a), we do not compute the new radius by the points' distance. So the stroke's points' radiuses remain the same as original. We obtain a better combination at the lion's jaw and front leg. With stroke translation, we can combine the fur and the stroke of the jaw better. In Figure 9(b), we do not use the stroke translation. It is because this drawing does not have overwriting strokes. We get better combinations at the left breast, the left sleeve, the creases of the pants and the waist.

Next, we compare our results with the results of Shesh and Chen [16]. In Figure 10(a), we obtain better results at the shoulder, the neck and the glasses. In Figure 10(b), we use segmentation to segment the long zigzag strokes as Shesh and Chen did, but we do not use the color attribute in the combination error. We use stroke translation to move the hatching together, so we can get a clear result that does not have hatching everywhere.

5. CONCLUSION

5.1 Summary

We have proposed a novel algorithm to simplify the line drawing. We calculate the tangent and curvature of the input strokes as stroke attributes. Then we segment the strokes based on the curvature and produce the strokes that are easier to be combined. After segmentation, the end points can have more appropriate tangent to present the orientation of the strokes. We apply the stroke translation to centralize the strokes. Then we draw the strokes to the texture and apply a low-pass Gaussian filter. The strokes are translated to the limited nearby position with the highest weight based on a weight map of the filtering results. Then we pair the strokes based on the end points' position and tangent. We calculate the end points' spatial and angular differences with other stroke's points. After checking the conditions of combination,

we pair and combine the strokes. The combination is ordered according to the weight of the strokes. At each combination step, we only deal with two strokes. We combine the pairs in an increasing order of the weight. By this simple algorithm, we can obtain a simplified line drawing.

5.2 Future Works

Although we can simplify the strokes with the proposed simple algorithm, we have to use many user-defined parameters. The users need to do the try and error for getting the best result. They need to know the meaning of each parameter. If we reduce the parameters, we can build a level-of-detail (LOD) structure. It should be a graph in which the points are the strokes and the edges are the error. Then we can combine the strokes from the smallest error edge and update the error after each stroke combination. With good data structure, we can reduce the computation time and obtain the better results by the better simplification order. Currently, we cannot simplify the strokes locally. With the density measure, we may control the local simplification. Then we can simplify a local area with different parameter and reduce the strokes.

REFERENCES

[1] P. Barla, J. Thollot, and F. X. Sillion. Geometric clustering for line drawing simplification. In Proc. Eurographics Symp. Rendering, pages 183--192, 2005.

[2] T. Baudel. A mark-based interaction paradigm for free-hand drawing. In Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology, pages 185--192, 1994.

[3] F. Cole, D. DeCarlo, A. Finkelstein, K. Kin, K. Morley, and A. Santella. Directing gaze in 3d models with stylized focus. In Eurographics Symposium on Rendering, volume 2, 2006.

[4] O. Deussen and T. Strothotte. Computer-generated pen-and-ink illustration of trees. In Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pages 13--18, 2000.

[5] Hongbo Fu, Shizhe Zhou, Ligang Liu, and Niloy Mitra. Animated construction of line drawings. ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA), 30(6):133, 2011.

[6] S. Grabli, F. Durand, and F.X. Sillion. Density measure for line-drawing simplification. In Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on, pages 309--318, 2004.

[7] A. Hertzmann. Stroke-based rendering. SIGGRAPH, 2002.

[8] T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: A technique for rapid geometric design. In Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology, pages 105--114, 1997.

[9] L.B. Kara, C.M. D'Eramo, and K. Shimada. Pen-based styling design of 3d geometry using concept sketches and template models. In Proceedings of the 2006 ACM Symposium Solid and Physical Modeling, volume 6, pages 149--160, 2006.

[10] G. Orbay and L.B. Kara. Beautification of design sketches using trainable stroke clustering and curve fitting. Visualization and Computer Graphics, IEEE Transactions on, 17(5):694--708, 2011.

[11] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, page 581, 2001.

[12] B. Preim and T. Strothotte. Tuning rendered line-drawings. In Proceedings of Winter School in Computer Graphics--The third Central European Conference on Computer Graphics' 95, pages 227--237, 1995.

[13] R. Pusch, F. Samavati, A. Nasri, and B. Wyvill. Improving the sketch-based interface. The Visual Computer, 23(9):955--962, 2007.

[14] P.L. Rosin. Grouping curved lines. In 5th British Machine Vision Conf, pages 265--274, 1994.

[15] E. Saund. Labeling of curvilinear structure across scales by token grouping. In Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on, pages 257--263, 1992.

[16] A. Shesh and B. Chen. Efficient and dynamic simplification of line drawings. Computer Graphics Forum, 27(2):537--545, 2008.

[17] B. Wilson and K.L. Ma. Rendering complexity in computer-generated pen-and-ink illustrations. In Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering, pages 129--137, 2004.