2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Sketch Simplification by Classifying Strokes

Toru Ogawa, Yusuke Matsui, Toshihiko Yamasaki and Kiyoharu Aizawa
Department of Information and Communication Engineering, The University of Tokyo, Tokyo, Japan

***Abstract*—In this paper, we propose a novel approach to creating clean line drawing from a scribbled sketch automatically. The main problem is determining which strokes of a scribbled sketch should be merged. We use a machine learning approach to solve this problem. Our method can automatically generate training data by comparing scribbled sketches with manually drawn line drawings without using annotations. In order to verify the generated training data, we merged strokes and created clean line drawings in accordance with the generated training data. In addition, we trained a support vector machine to estimate the pairs of strokes to be merged. Further, we verified that our method can create line drawings using this estimator.**

I. INTRODUCTION

Illustrations are widely used visual media that are designed to attract viewers; examples include creatures in games (Fig. 1a) and product mascots (Fig. 1b). Despite the high demand for commercial applications, the development of illustrations takes time because they are typically drawn by hand. Therefore, automating some of the steps involved in the drawing processes can reduce the burden on illustrators and help them to focus on developing more creative processes.

Drawing an illustration consists of three steps: rough sketching, line drawing, and colorization (Fig. 2). First, an illustrator briefly draws strokes to determine the position and shape of objects (Fig. 2a). Strokes in the rough sketch (*rough strokes*) are roughly drawn and contain visually unsuitable features such as broken and overtraced strokes (Fig. 3a, 3b). Therefore, rough strokes cannot be directly used in the finished work. Second, strokes without unsuitable features are drawn based on the rough strokes (Fig. 2b). Such *line strokes* are used to depict the final illustration. Finally, the illustrator applies colors, creating effects such as highlights and shadows (Fig. 2c).

Of the three steps above, we focus on generating a line drawing from a rough sketch. This process is accomplished by properly merging unsuitable strokes such as broken strokes and overtraced strokes (Fig. 3a, 3b) into a single line (Fig. 3c).

(a) Creature from game [1]
(b) Product mascot (Hatsune Miku, © Crypton Future Media, Inc. 2007, licensed under a CC BY-NC [2])

Fig. 1: Examples of illustrations.

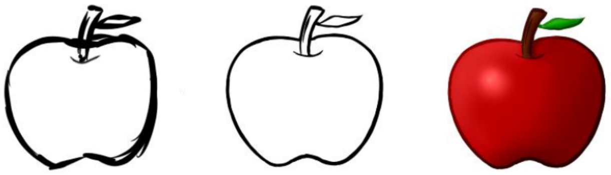

(a) rough sketch (b) line drawing (c) colorized image

Fig. 2: Three processes involved in drawing an illustration [3].

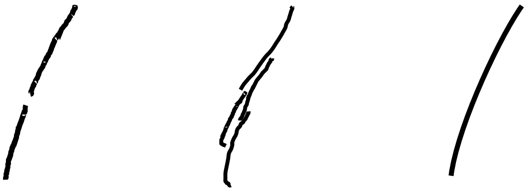
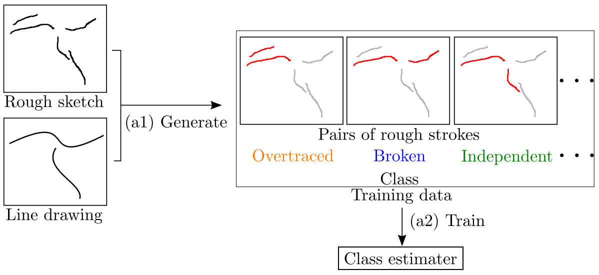

(a) broken (b) overtraced (c) merged stroke

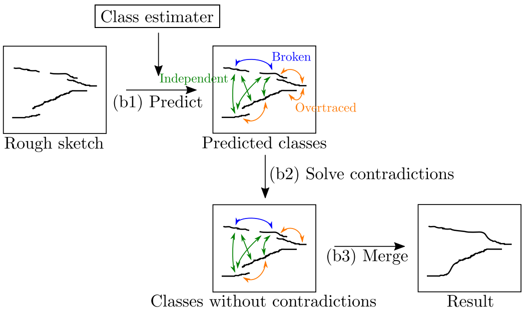
Fig. 3: Unsuitable features. (a) Broken: a stroke is separated into several short strokes along the original stroke. (b) Overtraced: several short, rough strokes construct a line stroke. (c) Merged stroke: gaps and overlapped strokes are merged into a single stroke in the line drawing.

The main approach is to use geometric features [4], [5], but these methods are sensitive to the parameters used (threshold values). It is difficult to determine the appropriate parameters for each rough sketch, e.g., some parameters that work well for rough sketches drawn by an illustrator may not work well for others. Another approach is to use machine learning [6], but this method requires manually annotated strokes for training, which are tedious and time consuming to construct.

In this paper, we propose a novel approach to cluster rough strokes and create a line drawing automatically. We use machine learning similar to [6]; however, our method can be trained without the use of manual annotations. Our method has two phases, namely training and simplification (Fig. 4). The training phase consists of two steps (Fig. 4a). First, given combinations of a rough sketch and a line drawing (Fig. 5), we detect correspondences from a rough sketch and a line drawing, and we classify each pair of rough strokes automatically (Fig. 4a1). Second, we train an estimator, which returns the connection type (class) from a pair of rough strokes (Fig. 4a2). The simplification phase consists of three steps (Fig. 4b) Given an input rough sketch, we first obtain two strokes from the rough sketch, and predict how to merge

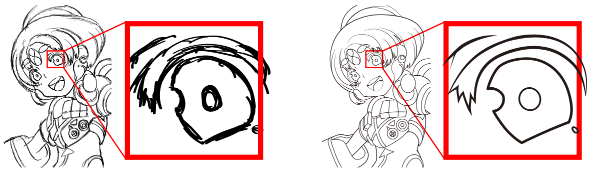


(a) train



(b) simplification

Fig. 4: Proposed method



(a) rough sketch (b) line drawing

Fig. 5: Combination of a rough sketch and a line drawing [7].

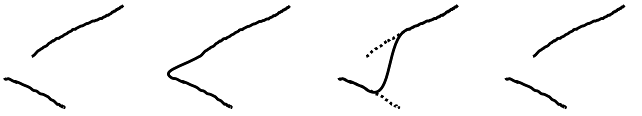
the strokes using the trained estimator (Fig. 4b1). Then, we solve contradictions (defined in Sec. III-E) in the predicted results (Fig. 4b2). Finally, we create a line drawing by merging rough strokes according to the predicted results (Fig. 4b3). The advantage of our approach is that we do not require any manual tuning steps (parameters, annotations, etc.) because we automatically train those steps using pairs. However, the disadvantage is that we must collect combinations of a rough sketch and a line drawing for the training step.

II. RELATED WORKS

Barla et al. [4] used ε-lines for grouping strokes. A ε-line is defined as a stroke that has no self-intersection. In their study, they group strokes if there exists a ε-line that covers all strokes in the group in the range of $\varepsilon/2$.

Liu et al. [8] used closures to group strokes, where a closure is defined as an area surrounded by strokes. Using three geometric features of stroke pairs, namely proximity, continuity, and parallelism, they determine whether a stroke pair should be merged. To improve the accuracy, they change the threshold value according to the size of the closure.

Orbay et al. [6] used machine learning to group strokes. They predict the connectivity of stroke pairs using a neural network. First, they compute three features, namely the remoteness, misalignment, and discontinuity for each pair of strokes. Second, they train a neural network, where the inputs are the three aforementioned stroke pair features, and the output is the connectivity of the pair. Training data are manually clustered sketches. In this step, features are normalized by the length (the number of strokes) of the shortest path between strokes. This normalization reduces the distance between the same cluster's strokes over the given distance. Finally, they compute features for stroke pairs of unclustered sketches, and cluster them using the trained neural network.



(a) input (b) “Broken” (c) “Overtraced” (d) “Independent”

Fig. 6: Classes for stroke pairs.

III. PROPOSED METHOD

In Sec. III-A, we explain the data format of the input illustrations and preprocessing, while in Sec. III-B, we define the classes of stroke pairs. In Sec. III-C and Sec. III-D, we show how to train a class estimator from pairs of rough sketches and line drawings (Fig. 4a). In Sec. III-E and Sec. III-F, we show how to simplify a rough sketch using the trained class estimator and obtain a line drawing (Fig. 4b).

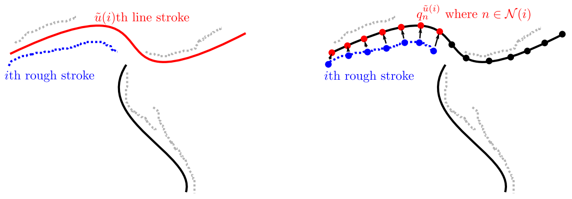
A. *Data Format*

In our proposed method, we assume that both rough sketches and line drawings are vector graphics; this means that each stroke in the input images is a sequence of 2D points. If the input images are raster graphics, the images are vectorized in advance (e.g. by [9]). In this study, we used vector graphics consisting of cubic Bezier curves. We converted a cubic Bezier curve to a sequence of 2D points by sampling points with a constant interval (1 px).

In addition, we assume that each stroke is smooth. If a stroke has n bending points, we divide the stroke into $n + 1$ short strokes. A bending point is defined as the point at which the curvature of the stroke exceeds a constant threshold, which is fixed as 1.0 in this study.

B. *Stroke Pairs Class*

To simplify a rough sketch, we focus each stroke pair (two rough strokes) in the input rough sketch. Given two input rough strokes, we define three classes, namely “Broken,” “Overtraced,” and “Independent,” as shown in Fig. 6. “Broken” means that the two strokes should be connected as a single stroke by filling the gap (Fig. 6b). “Overtraced” means that the two strokes should be merged by mixing overlapping sections smoothly (Fig. 6c). “Independent” means that the two strokes should not be connected (Fig. 6d). As discussed later in Sec. III-D, we train an estimator that takes two rough strokes as inputs and predicts one of the three classes.


$\tilde{u}(i)$th line stroke
ith rough stroke
$q_n^{\tilde{u}(i)}$ where $n \in \mathcal{N}(i)$
ith rough stroke

(a) corresponding stroke (b) corresponding points

Fig. 7: Corresponding stroke and points. Dash: rough stroke. Solid: line stroke. Blue: focused rough stroke (ith rough stroke). Red: corresponding stroke ($\tilde{u}(i)$th line stroke) / corresponding points ($q_n^{\tilde{u}(i)}$, where $n \in \mathcal{N}(i)$)

C. Generating Training Data

Next, we explain how to create training data correspondences (Fig. 4a1). Given a line drawing and a rough sketch without any manual annotations, we find the correspondences of the line strokes and rough strokes. We denote the ith rough stroke as a sequence of 2D points: $\{\boldsymbol{p}_m^i \in \mathbb{R}^2 | m = 1, 2, \dots\}$. In the same manner, we define the uth line stroke as: $\{\boldsymbol{q}_n^u \in \mathbb{R}^2 | n = 1, 2, \dots\}$. First, we find a corresponding stroke for each rough stroke (Fig. 7a). A stroke corresponding to the ith rough stroke is defined as a line stroke that is the nearest to the ith rough stroke. Therefore, if the $\tilde{u}(i)$th line stroke corresponds to the ith rough stroke, $\tilde{u}(i)$ is defined as:

$$\tilde{u}(i) = \operatorname*{argmin}_u d(i, u) \tag{1}$$

$d(i, u)$ is the distance from the ith rough stroke to the uth line stroke, which is defined as:

$$d(i, u) = \sum_m \min_n \|\boldsymbol{p}_m^i - \boldsymbol{q}_n^u\| \tag{2}$$

By creating a distance map $d^u(\boldsymbol{p})$ from the uth line stroke, we can compute $d(i, u)$ efficiently [10]. We need to compute the distance map only once for each line stroke, because $d^u(\boldsymbol{p})$ does not depend on rough strokes. In addition, we use the K-nearest heuristic in order to reduce the number of line strokes to compute $d(i, u)$. We compute the distance between the line strokes and the centers of mass of the rough strokes. We use the K-nearest line strokes to compute $d(i, u)$. In this study, we use $K = 5$.

Second, we compute the point-wise correspondences (Fig. 7b). For each rough stroke, the corresponding line stroke is found using Eq. 1. Then, for each point in the rough stroke, we find the corresponding points from the corresponding line stroke. Considering the ith rough stroke, the identifiers of the corresponding points, $\mathcal{N}(i)$, are defined as:

$$\mathcal{N}(i) = \left\{ \operatorname*{argmin}_n \left\| \boldsymbol{p}_m^i - \boldsymbol{q}_n^{\tilde{u}(i)} \right\| \,\middle|\, m = 1, 2, \dots \right\} \tag{3}$$

Finally, we classify the rough stroke pairs into three classes (“Broken,” “Overtraced,” and “Independent”). In order to reduce the number of pairs, we use pairs whose distance is less than a constant value d, which is fixed as 25 in this study.

If two rough strokes are placed far apart, it is likely that they correspond to different line strokes. We classify a stroke pair (i, j) according to the corresponding stroke $(\tilde{u}(i), \tilde{u}(j))$ and identifiers $(\mathcal{N}(i), \mathcal{N}(j))$. If the two strokes correspond to different line strokes $(\tilde{u}(i) \neq \tilde{u}(j))$, we classify the pair as “Independent” (Fig. 8a). Otherwise, if the points corresponding to the two strokes do not intersect $(\mathcal{N}(i) \cap \mathcal{N}(j) = \phi)$, we classify the pair as “Broken” (Fig. 8b). If the points intersect $(\mathcal{N}(i) \cap \mathcal{N}(j) \neq \phi)$, we classify the pair as “Overtraced” (Fig. 8c).

D. Training the Estimator

In the previous process, we created training data, which is a set of rough stroke pairs with classes. We train an estimator that predicts a class from a pair of rough strokes (Fig. 4a2). We employed a standard support vector machine (SVM) [11] with a radial basis function (RBF) kernel.

The input for the SVM is a 32-dimensional feature vector, which is computed from the two rough strokes as follows. First, to make the vector position-, rotation-, and scale-invariant, we regularize the coordinates of each point of the rough strokes. Let us define $\boldsymbol{g} = (g_x, g_y)^\top$ as the center of mass of the longer stroke, and $\boldsymbol{g'} = (g'_x, g'_y)^\top$ for the shorter stroke. Each point $\boldsymbol{p}$ in the stroke is mapped to $\bar{\boldsymbol{p}}$, which is denoted as:

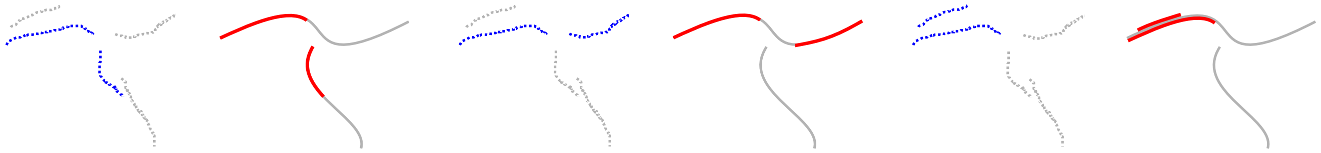
$$\bar{\boldsymbol{p}} = \frac{1}{\|\boldsymbol{g} - \boldsymbol{g'}\|^2} \begin{pmatrix} g'_x - g_x & g'_y - g_y \\ -(g'_y - g_y) & g'_x - g_x \end{pmatrix} (\boldsymbol{p} - \boldsymbol{g}). \tag{4}$$

Second, we select eight points from each stroke at even intervals. Finally, the x and y coordinates of the 16 selected points are concatenated in order to form a 32-dimensional vector.

E. Prediction

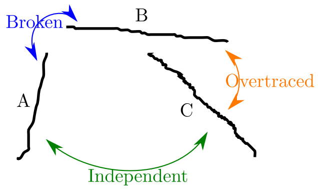
First, the class of each pair of rough strokes is predicted by the estimator (Fig. 4b1). There may exist some contradictions among the outputs of the estimator. A contradiction is defined as a state in which it is not possible to satisfy all of the predicted classes. For example, if we predict classes for three stroke pairs, (A, B), (B, C), and (C, A), where the results are “Broken,” “Overtraced,” and “Independent,” respectively (Fig. 9a), these results have a contradiction. If we merge A, B, and C into a single line stroke, the stroke $(\{A, B, C\})$ is in conflict with the “Independent” class of (C, A). (Fig. 9b). If we keep C as a single stroke, and merge only A and B, the resultant $(\{A, B\}, \{C\})$ is in conflict with the “Overtraced” class of (B, C) (Fig. 9c).

We solve these contradictions before the merging step (Fig. 4b2). In order to solve contradictions, we use a greedy method to determine the prediction results that should be ignored. We divide rough strokes $(1, 2, \dots, N)$ into sets $(\mathcal{S}_1, \mathcal{S}_2, \dots)$ that satisfy the predicted results as much as possible. Note that N is the total number of rough strokes, and $\mathcal{S}_i$ is a set of strokes that satisfies $\bigcup \mathcal{S}_i = \{1, 2, \dots, N\}$

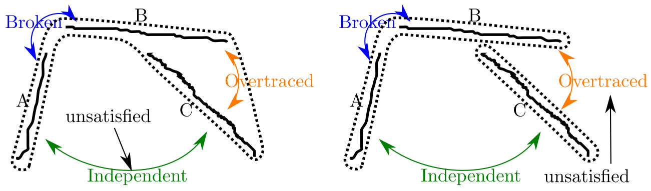


(a) "Independent"　(b) "Broken"　(c) "Overtraced"

Fig. 8: Classification of stroke pairs according to correspondence (left: rough strokes, right: line strokes).



(a) prediction results



(b) $\{A, B, C\}$　(c) $\{A, B\}, \{C\}$

Fig. 9: Contradiction in the prediction results.

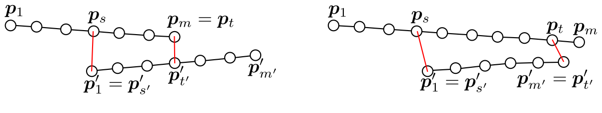
with $\mathcal{S}_i \cap \mathcal{S}_j = \phi$. In order to find such sets, we minimize the cost E, which is defined as:

$$E = \sum_{(i,j)\in\mathcal{L}} -\delta(i,j) + \sum_{(i,j)\in\bar{\mathcal{L}}} \delta(i,j) \tag{5}$$

where $\mathcal{L}$ is a set of stroke pairs predicted to be "Broken" or "Overtraced," and $\bar{\mathcal{L}}$ is a set of stroke pairs predicted to be "Independent." $\delta(i,j)$ returns 1 if two strokes i and j belong to the same set; it returns 0 otherwise. We minimize E by performing the following steps.

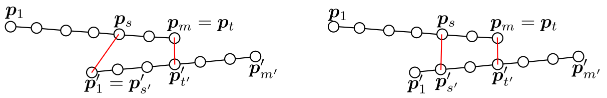
1) Initialize clusters as $\mathcal{S}_1 = \{1\}$, $\mathcal{S}_2 = \{2\}$, $\ldots$, $\mathcal{S}_N = \{N\}$.
2) Find a pair of clusters $(\mathcal{S}_i, \mathcal{S}_j)$ that minimize $\Delta E(\mathcal{S}_i, \mathcal{S}_j) = |\mathcal{L} \cap (\mathcal{S}_i \times \mathcal{S}_j)| - |\bar{\mathcal{L}} \cap (\mathcal{S}_i \times \mathcal{S}_j)|$.
3) Finish the process if $\Delta E(\mathcal{S}_i, \mathcal{S}_j) \geq 0$.
4) Merge $\mathcal{S}_i$ and $\mathcal{S}_j$.
5) Back to 2).

Finally, we remove the prediction result for each pair (i,j) such that $(i,j) \in \mathcal{L} \wedge \delta(i,j) = 0$ or $(i,j) \in \bar{\mathcal{L}} \wedge \delta(i,j) = 1$.

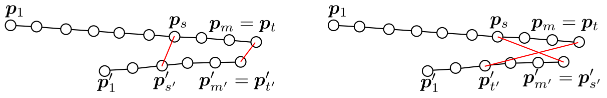

$\boldsymbol{p}_1$
$\boldsymbol{p}_s$
$\boldsymbol{p}_m = \boldsymbol{p}_t$
$\boldsymbol{p}'_1 = \boldsymbol{p}'_{s'}$
$\boldsymbol{p}'_{t'}$
$\boldsymbol{p}'_{m'}$
$\boldsymbol{p}_1$
$\boldsymbol{p}_s$
$\boldsymbol{p}_t$
$\boldsymbol{p}_m$
$\boldsymbol{p}'_1 = \boldsymbol{p}'_{s'}$
$\boldsymbol{p}'_{m'} = \boldsymbol{p}'_{t'}$

(a)　(b)

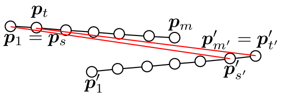
Fig. 10: Overlapping section



(a) condition 1　(b) condition 2


$\boldsymbol{p}_1$
$\boldsymbol{p}_s$
$\boldsymbol{p}_m = \boldsymbol{p}_t$
$\boldsymbol{p}'_1$
$\boldsymbol{p}'_{s'}$
$\boldsymbol{p}'_{m'} = \boldsymbol{p}'_{t'}$
$\boldsymbol{p}_1$
$\boldsymbol{p}_s$
$\boldsymbol{p}_m = \boldsymbol{p}_t$
$\boldsymbol{p}'_1$
$\boldsymbol{p}'_{t'}$
$\boldsymbol{p}'_{m'} = \boldsymbol{p}'_{s'}$

(c) condition 3　(d) condition 4


$\boldsymbol{p}_t$
$\boldsymbol{p}_1 = \boldsymbol{p}_s$
$\boldsymbol{p}_m$
$\boldsymbol{p}'_{m'} = \boldsymbol{p}'_{t'}$
$\boldsymbol{p}'_{s'}$
$\boldsymbol{p}'_1$

(e) condition 5

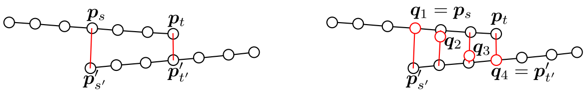
Fig. 11: Examples of undesired candidates (captions show unsatisfied contradiction).

F. Merging strokes

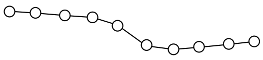
After the contradictions are solved, we merge the rough strokes (Fig. 4b3). First, we merge the stroke pairs that are predicted to be "Overtraced." We merge an "Overtraced" pair by interpolating the overlapping section. For simplicity, we denote the stroke pair as $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_m\}$ and $\{\boldsymbol{p}'_1, \boldsymbol{p}'_2, \ldots, \boldsymbol{p}'_{m'}\}$. The overlapping section is defined as the range $[s,t]$ and $[s',t']$, where $s,t \in [1,m]$ and $s',t' \in [1,m']$. The overlapping section must satisfy the following criteria:

1) $|s-t| = |s'-t'|$.
2) Two points of $\boldsymbol{p}_s, \boldsymbol{p}_t, \boldsymbol{p}'_{s'}, \boldsymbol{p}'_{t'}$ are the end points of strokes, and the other two points are not the end points.
3) If $\boldsymbol{p}_s$ is the end point of a stroke, $\boldsymbol{p}'_{s'}$ is not the end point (t, t' also have the same relation).
4) Two line segments, $\boldsymbol{p}_s - \boldsymbol{p}'_{s'}$ and $\boldsymbol{p}_t - \boldsymbol{p}'_{t'}$, do not cross.
5) Minimize $\frac{1}{|s-t|}(\|\boldsymbol{p}_s - \boldsymbol{p}'_{s'}\| + \|\boldsymbol{p}_t - \boldsymbol{p}'_{t'}\|)$.

Fig. 10 shows overlapping sections that satisfy the above five conditions. Fig. 11 shows undesired candidates of overlapping sections. We merge "Overtraced" pairs in ascending order of $\frac{1}{|s-t|}(\|\boldsymbol{p}_s - \boldsymbol{p}'_{s'}\| + \|\boldsymbol{p}_t - \boldsymbol{p}'_{t'}\|)$. If the overlapping section is shorter than any stroke of the stroke pair (Fig. 12a), we compute new points by mixing two strokes. The mixing weights are changed according to the square of the distance from the end points of strokes (Fig. 12b, 12c). The new points

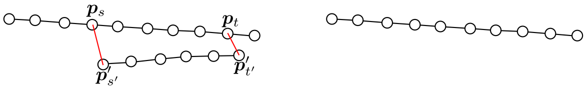


(a) input (b) interpolation



(c) result

Fig. 12: Merging the "Overtraced" pair (the overlapping section is shorter than any of the two strokes).



(a) input (b) result

Fig. 13: Merging the "Overtraced" pair (the overlapping section is the same as one of the two strokes).

$\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{|s-t|+1}$ are defined as:

$$\boldsymbol{q}_i = \frac{(1-u)^2 \boldsymbol{p}_{i+s-1} + u^2 \boldsymbol{p}'_{i+s'-1}}{(1-u)^2 + u^2} \tag{6}$$

where $\boldsymbol{p}_t$ and $\boldsymbol{p}'_{s'}$ are the end points of strokes, and u is defined as $\frac{i-1}{|s-t|}$. If the overlapping section is as long as one of the two strokes (Fig. 13a), we simply remove the shorter stroke (Fig. 13b).

Second, we merge stroke pairs that are predicted to be "Broken." We interpolate the gap between the nearest end points of strokes with a cubic Bezier curve [12] (Fig. 14). The cubic Bezier curve $\boldsymbol{b}(t)$ is defined as:

$$\boldsymbol{b}(t) = (1-t)^3 \boldsymbol{p} + 3t(1-t)^2 \boldsymbol{c} + 3t^2(1-t)\boldsymbol{c}' + t^3 \boldsymbol{p}' (t \in [0,1])$$

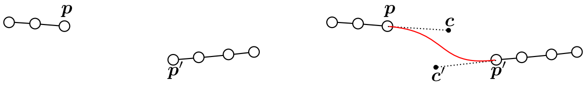
where $\boldsymbol{p}, \boldsymbol{c}, \boldsymbol{c}', \boldsymbol{p}'$ are control points. We use the end points of the strokes as $\boldsymbol{p}$ and $\boldsymbol{p}'$. Then, we compute $\boldsymbol{c}$ and $\boldsymbol{c}'$ using the following conditions:

1) $\boldsymbol{c} - \boldsymbol{p}$ is parallel to the tangent vector of the stroke at $\boldsymbol{p}$ $\boldsymbol{c}' - \boldsymbol{p}'$ (the same for $\boldsymbol{p}'$ and $\boldsymbol{c}'$).
2) $\|\boldsymbol{c} - \boldsymbol{p}\| = \|\boldsymbol{c}' - \boldsymbol{p}'\| = \frac{1}{2}\|\boldsymbol{p} - \boldsymbol{p}'\|$.

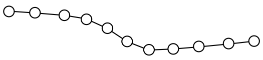
Similar to the case with the "Overtraced" pairs, we merge "Broken" pairs in ascending order of the distance between the end points of the strokes ($\|\boldsymbol{p} - \boldsymbol{p}'\|$).

IV. EXPERIMENTS

We conducted two experiments: simplification using correct classes and simplification using predicted classes. The dataset for the experiments was provided by MUGENUP Inc. [7]. This dataset consists of 10 combinations of a rough sketch (Fig. 15a) and a line drawing (Fig. 15b). All of the illustrations are vector images, and were drawn by a professional illustrator.

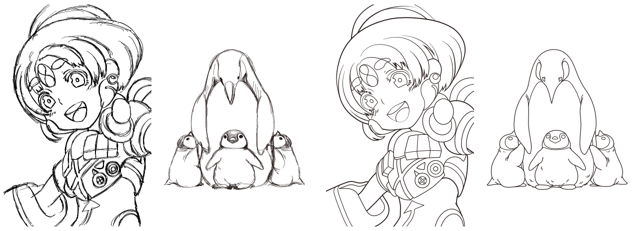


(a) input (b) interpolation



(c) result

Fig. 14: Merging the "Broken" pair.



(a) rough sketches (b) line drawings

Fig. 15: Dataset [7].

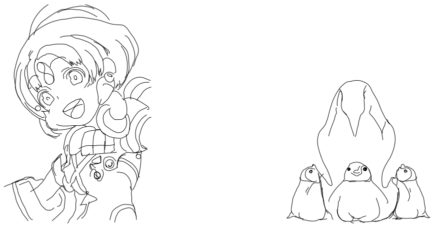
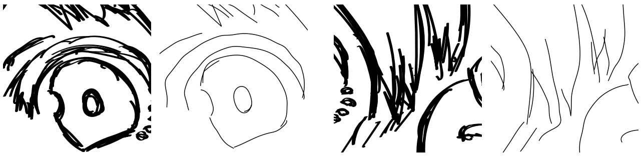


Fig. 16: Results of merging.

A. Simplification using Training Data

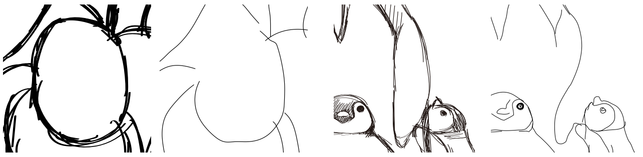
In this experiment, we tested both parts of our method, generating training data and merging strokes. We generated training data using the combination of a rough sketch and a line drawing, and we merged the strokes of the same rough sketch using generated training data. In this experiment, we skipped two steps: training the estimator (Fig. 4a2) and using the estimator for prediction (Fig. 4b1). In other words, the results show how our method would behave if the prediction step works perfectly. This experiment is useful for separately evaluating the prediction part and other parts.

Fig. 16 shows the results of merging. Complex regions such as the eye and hair can be merged properly (Fig. 17). On the other hand, U-structures such as ovals (Fig. 18a) and wings (Fig. 18b) were not successfully merged during the merging process. The U-structures were broken because of the ambiguity of the overlapping section. Given a U-structure, there are two candidates for the overlapping section (Fig. 19). In our method, we always choose pattern α (Fig. 19b), and we merge all of the strokes. In this manner, the U-structures are broken. It is difficult to correctly decide between patterns α and β, and this remains our future work.



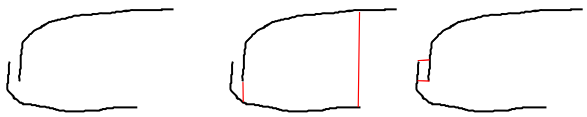
(a) eye (b) hair

Fig. 17: Complex regions (left: input, right: result).



(a) oval (b) wing

Fig. 18: U-structures (left: input, right: result).



(a) input (b) pattern α (c) pattern β

Fig. 19: Overlapping section of U-structure.

TABLE I: Percentage of pairs removed by solving contradictions step (I, B, and O stand for “Independent,” “Broken,” and “Overtraced”).

		Predicted	
		I	B, O
Training Data	I	13.5%	40.7%
	B, O	48.8%	15.3%

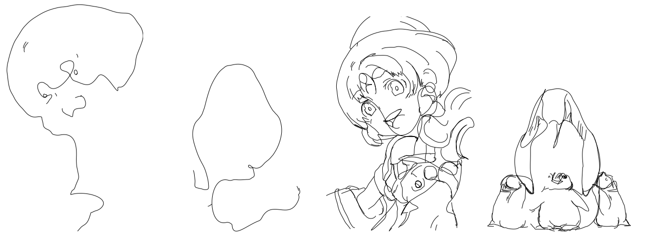
B. Simplification using Predicted Classes

In this experiment, we merged strokes under two conditions: using prediction results before solving contradictions and using results after solving contradictions. We split the dataset into nine training combinations and one test combination. The training data consists of 72076 “Independent” pairs, 8715 “Broken” pairs, and 32692 “Overtraced” pairs.

TABLE I shows the percentage of pairs that were removed by solving contradictions. The contradiction-solving step removed more incorrect pairs than correct pairs. Fig. 20 shows the results of the merging step. If we merge the strokes using the SVM outputs before solving contradictions, most of the rough strokes are merged, and the shapes of the objects are significantly changed. After solving contradictions, the merged pairs were regulated and the shapes of the objects were retained. On the other hand, we successfully merged strokes that were used to construct broken strokes or overtraced strokes in several regions.

V. CONCLUSION

In this study, we proposed a method to convert a rough sketch into a line drawing using a machine learning approach. Our method has two advantages. First, in the training phase, our method generates training data automatically from combinations of a rough sketch and a line drawing. We evaluated this step using generated training data as the input of the merging step. In most cases, our method created good line drawings. Second, after obtaining a prediction using the estimator, our method solved contradictions by removing a portion of the predicted classes. We performed tests to verify that this step was more likely to remove incorrect pairs than correct pairs, and the line drawings that were created using the contradiction-solving step were better than those created without the step.



(a) using raw prediction results (b) using result without contradictions

Fig. 20: Merged rough sketch obtained using predicted classes.

ACKNOWLEDGMENT

This work was supported by the Strategic Information and Communications R&D Promotion Programme (SCOPE). We would like to thank MUGENUP Inc. [7] for providing us with the illustration dataset.

REFERENCES

[1] JewelSaviorFREE, http://www.jewel-s.jp/.

[2] Creative Commons, http://creativecommons.org/licenses/by-nc/3.0/.

[3] Y. Chien, W.-C. Lin, T.-S. Huang, and J.-H. Chuang, “Line drawing simplification by stroke translation and combination,” in *Fifth International Conference on Graphic and Image Processing*. International Society for Optics and Photonics, 2014, pp. 90 690X–90 690X.

[4] P. Barla, J. Thollot, and F. X. Sillion, “Geometric clustering for line drawing simplification,” in *ACM SIGGRAPH 2005 Sketches*. ACM, 2005, p. 96.

[5] S. Wang, S. Qin, and M. Gao, “New grouping and fitting methods for interactive overtraced sketches,” *The Visual Computer*, vol. 30, no. 3, pp. 285–297, 2014.

[6] G. Orbay and L. B. Kara, “Beautification of design sketches using trainable stroke clustering and curve fitting,” *Visualization and Computer Graphics, IEEE Transactions on*, vol. 17, no. 5, pp. 694–708, 2011.

[7] MUGENUP Inc., http://mugenup.com/.

[8] X. Liu, T. Wong, and P. Heng, “Closure-aware sketch simplification,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 168, 2015.

[9] G. Noris, A. Hornung, R. W. Sumner, M. Simmons, and M. Gross, “Topology-driven vectorization of clean line drawings,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, p. 4, 2013.

[10] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, “Parametric correspondence and chamfer matching: Two new techniques for image matching,” DTIC Document, Tech. Rep., 1977.

[11] J. A. Suykens and J. Vandewalle, “Least squares support vector machine classifiers,” *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[12] L. Cinque, S. Levialdi, and A. Malizia, “Shape description using cubic polynomial bezier curves,” *Pattern Recognition Letters*, vol. 19, no. 9, pp. 821–828, 1998.