An interactive system for structure-based ASCII art creation

Katsunori Miyake† Henry Johan‡ Tomoyuki Nishita†

†The University of Tokyo ‡Nanyang Technological University

Abstract

Non-Photorealistic Rendering (NPR), whose aim is to create artistic style images, is one of the important research topics in computer graphics. One example of NPR is an art form called ASCII art which represents pictures using character strings. ASCII art is commonly used in media that cannot display images or mainly use text, such as e-mail and bulletin board system. ASCII art can be categorized into two styles, tone-based style and structure-based style. The structure-based style can represent content by using less number of ASCII characters compared to the tone-based style. However, in general, it takes beginners a long time to create structure-based ASCII art. As such, an automatic method that creates structure-based ASCII art is required to reduce the tasks of ASCII art creation.

In this paper, we propose an interactive system which creates structure-based ASCII art. We provide four matching metrics for converting an image into an ASCII art.Based on experimental results, we found that the suitable matching metrics to produce visually pleasant ASCII art images depend on the type of the input images. Since our system can produce ASCII art images in a few seconds, users can create several ASCII art images using different matching metrics, then select the one they like the most.

1 Introduction

ASCII art is one type of artistic style images. It represents visual information using character strings which consist of 95 ASCII printable characters. ASCII is a character-encoding scheme, and is the acronym of American Standard Code for Information Interchange. ASCII art does not focus on photorealism. ASCII art is a popular art in cyberspace. We can see many ASCII art in various media which mainly use text, such as email and BBS on the Web. In Japanese BBS, we can see many ASCII art that represent outline of animation characters.

ASCII art can be categorized into two styles, tone-based style and structure-based style (Figure 1). Tone-based ASCII art uses the density of glyph to represent the intensity distribution of content. Structure-based ASCII art uses the line of glyph to represent the line structure of content, and can represent content by using less number of ASCII characters than tone-based ASCII art. As a result, structure-based ASCII art is often used in email and BBS. Most of these ASCII art are made manually. However, it is difficult for beginners to create structure-based ASCII art. As such, we propose an interactive system for creating structure-based ASCII art.

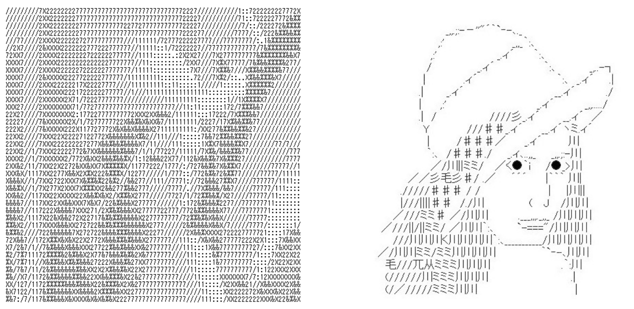

(a) Tone-based style (b) Structure-based style

Figure 1: *Examples of two styles.*

Our ASCII art creation framework consists of a preprocessing step and an ASCII art creation step. In the preprocessing step, we prepare glyph images of ASCII printable characters by rasterizing an outline font. The glyph images are of the same size. For ASCII art creation, the input images of our system are black-and-white images where the background is white and the content is black. Our system also allows users to perform drawing. To create the ASCII art image of an input image, we first divide the input image into grids which are of the same size as the glyph images. We call these grids as blocks. Then, we perform glyph matching which is a process of calculating the similarity between the blocks of the input image and glyph images. For each block, we put the best matching glyph in it. We perform this process for every block of the input image.

For block and glyph matching, we apply four matching metrics. The first one is template matching where we calculate the difference between the values of same position pixels in the block and glyph image, then we calculate the dissimilarity by using the Sum of Absolute Difference. The second metric is normalized cross-correlation where we calculate the similarity between the histograms of the distribution of black pixels. The third one is Histogram of Oriented Gradients (HOG) where we first determine HOG by calculating the gradient at each pixel, then we calculate difference between two HOGs. The last metric is distance transformation where we calculate the distance transformation image of the block and measure the distance of the glyph image to the distance transformed image.

Results of template matching and NCC are good if lines of input images are thick. Results of HOG can represent line directions. Results of distance transformation can represent line positions.

2 Related Work

There are some existing software programs for generating ASCII art, such as AA-lib [1]. Most of these programs create the tone-based ASCII art using a half-toning method. This method represents input images using the line density pattern of glyphs. Results of AA-lib are similar to half-toning results. However, AA-lib uses varying brightness font.

O ' Grady and Rickard [2] proposed a method to convert binary images into tone-based ASCII art using Non-negative Matrix Factorization. This method which uses Non-negative Matrix Factorization minimizes the difference between input images and glyphs in a pixel-by-pixel manner. The computational cost of this method is high.

Xu et al. [3] presented a method to create structure-based ASCII art. This method uses alignment-insensitive shape similarity metric which is similar to shape context, and uses deformation by adjusting the vertex positions of image polylines. To tolerate misalignment, this method used a histogram of a log-polar diagram. However, it is high computational cost to optimize polylines of input image.

3 User Interface

A purpose of our system is the creation of ASCII art from input images or drawn images by users in real-time. Users draw an illustration in the left area (Figure 2). Then, our system shows the result of ASCII art as text data in the right area. Users can also change the glyph of the results by choosing from displayed candidates.

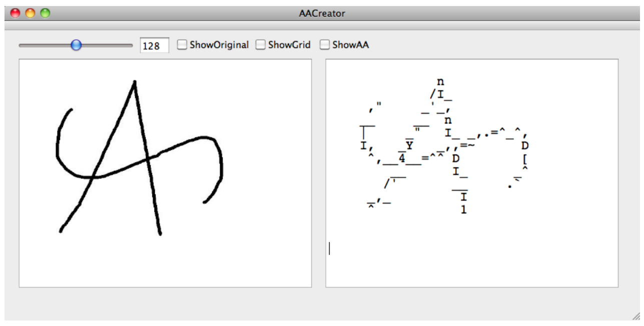


Figure 2: *The user interface of the proposed system. The left area is the drawing area. The right area displays the result of ASCII art.*

4 Matching Method

We describe the outline of glyph matching. Glyph matching is the process of matching which compares an input image with glyph images. The input of our method is a raster image. First, the input image is subdivided into grids which are of the same size as the glyph images. In this paper, these grids are called blocks. Second, we execute glyph matching which calculates similarity measure or dissimilarly measure between the input image and glyph images at each block. We set the glyph which is considered the best glyph by matching a block as one glyph of ASCII art string. We execute this process for every block. We describe four methods which are used in the matching process in the following sections.

4.1 Template Matching

We apply template matching to our method for calculating dissimilarly measure between a block of an input image and glyph images. This method considers pixel positions of images, while half-toning method considers the average value of brightness per block. Therefore, template matching is considered as a method that can represent the line feature of input image. We calculate dissimilarly measure by using Sum of Absolute Difference (SAD). SAD between a block and a glyph, f_{diff} is calculated using the following equation.

$$f_{diff} = \sum_{j}^{height} \sum_{i}^{width} |I(i,j) - G(i,j)|, \tag{1}$$

where I and G are the brightness values of a block of an input image and a glyph image, i and j are the coordinates of pixel position. $height$ and $width$ are the sizes of blocks. We calculate f_{diff} using every glyph images at the block. The glyph whose value of SAD is the lowest in glyph is set as a glyph of ASCII art string.

4.2 NCC with Histogram of Image Grid

Template matching using SAD is influenced by the difference between an input image and glyph images in line width. In this section, we introduce a method minimizing the influences of difference in line width. We also describe Normalized Cross-Correlation (NCC) which calculates similarity measure, while SAD represents dissimilarly measure.

To minimize the influence of the difference of the line width, we make a histogram of an image block. This histogram represents the number of black pixels in a block. To make a histogram of an image block, we divide the block using small grids (Figure 3). In each grid, we count the number of black pixels, and set the number in the same grid of the histogram.

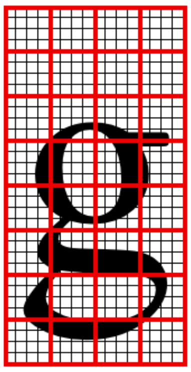


Figure 3: *Image block: Black grids are pixels, and red grids are clusters of pixels for histogram counting.*

NCC [4] is a method that calculates similarity measure. NCC between a block and a glyph image, f_{sim} is calculated using the following equation.

$$f_{sim} = \frac{\sum_{i}^{grids} H_{block}(i) H_{glyph}(i)}{\sqrt{\sum_{i}^{grids} H_{block}(i)^2 \times \sum_{i}^{grids} H_{glyph}(i)^2}}, \tag{2}$$

where H_{block} and H_{glyph} are the histograms of a block of an input image and a glyph image. $grids$ represent the set of grids in a block. Calculating the NCC value of every glyph image, the glyph whose f_{sim} is the highest is regarded as the best matching grid for the block. The closer f_{sim} is to 1, the more similar are the two histograms.

4.3 Histogram of Oriented Gradients

In this section, we introduce the computation of Histogram of Oriented Gradients (HOG) [5] , and describe a matching method using HOG. First, we calculate gradient magnitude, $m(i,j)$ and orientation, $\theta(i,j)$ at each pixel of images using the following equations.

$$m(i,j) = \sqrt{f_x(i,j)^2 + f_y(i,j)^2}, \tag{3}$$

$$\theta(i,j) = \tan^{-1}\left(\frac{f_y(i,j)}{f_x(i,j)}\right), \tag{4}$$

where f_x and f_y are the first order derivatives of horizontal and vertical directions. After calculation, we divide images by 5x5 pixels grid called cell. We create an orientation histogram which consists of nine cells. We make an orientation vector by using HOG blocks which consist of 3x3 cells (Figure 4), and we normalize the vector.

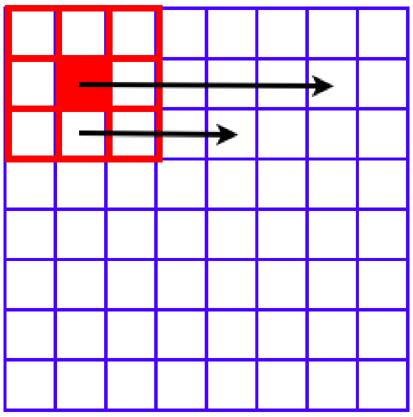


Figure 4: *HOG block consists of 3x3 cells. For calculating an orientation histogram at each HOG block, the center cell of HOG block is moving to the next cell in the direction of the arrows.*

We make a feature vector from all HOG blocks. If there is n HOG blocks in an image, the feature vector of HOG has $9n$ dimensions. The difference between HOGs , D_{HOG} is calculated using the following equation.

$$D_{HOG} = \sum_{i}^{Dimension} (H_i - H_g)^2, \tag{5}$$

where H_i and H_g are HOGs of block of an input image and a glyph image. We set the glyph which has the smallest mean square error as a glyph of ASCII art.

4.4 Distance Transformation

In this section, we describe a matching method using distance transform. Distance transformation is a method which creates the distance map from input images (Figure 5). The pixel value in a distance map is defined as the distance between the pixel and the nearest pixel of 0 value in the input image. We calculate the distance between a block of input image and glyph images using distance transformation. We calculate the distance between a block and glyph image, D_{ig} using the following equations.

$$D_{ig} = \sum_{j}^{height} \sum_{i}^{width} I_{DT}(i,j) G(i,j), \tag{6}$$

where I_{DT} is the value of the distance map of the block, and G is the brightness value of glyph image. We define the glyph whose distance is the smallest in glyphs as the glyph of ASCII art.

0	0	0	0	0	0
0	1	1	1	1	0
0	1	1	1	1	0
0	1	1	1	1	0
0	1	1	1	1	0
0	0	0	0	0	0

(a) Input image

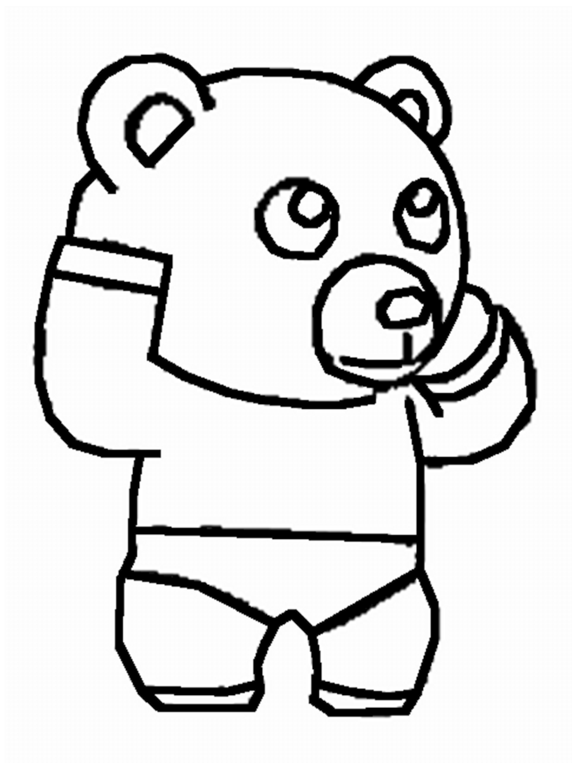
0	0	0	0	0	0
0	1	1	1	1	0
0	1	2	2	1	0
0	1	2	2	1	0
0	1	1	1	1	0
0	0	0	0	0	0

(b) Distance map

Figure 5: *Examples of distance transformation.*

5 Results

We show the results of our system. Our system is implemented using the C language and using the OpenCV library [6]. The experiments shown in this paper were conducted using a PC with an Intel Core 2 Duo 2.0 GHz CPU with 2.0GB main memory, and an NVIDIA GeForce 9400M GPU. We show the input images and the results using the matching methods in Figures 6 to 10. The resolution of the Bear image is 720x960 pixels. The resolution of the Shapes image is 512x528pixels. The sizes of the glyphs are 16x32pixel.


(a) Bear

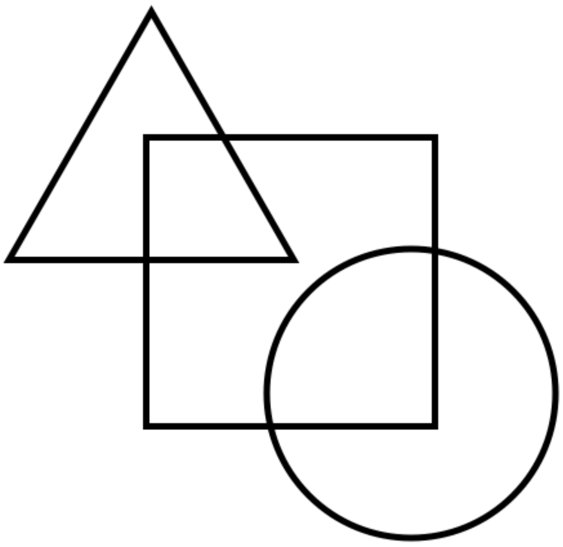

(b) Shapes

Figure 6: *Input images.*

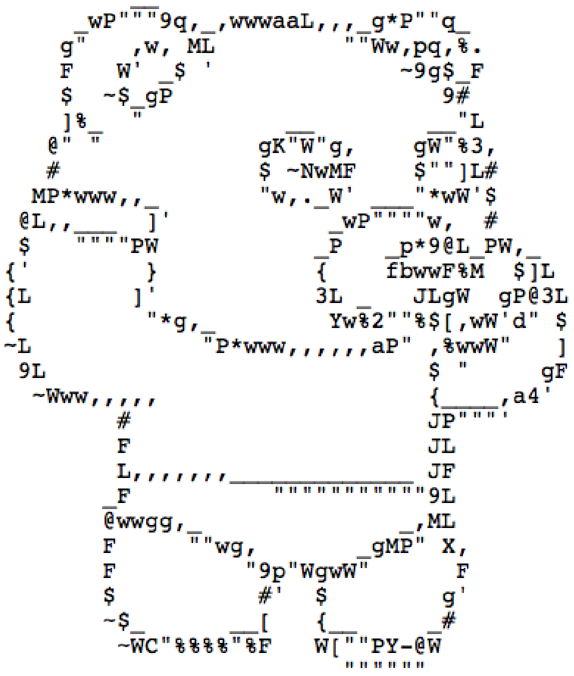
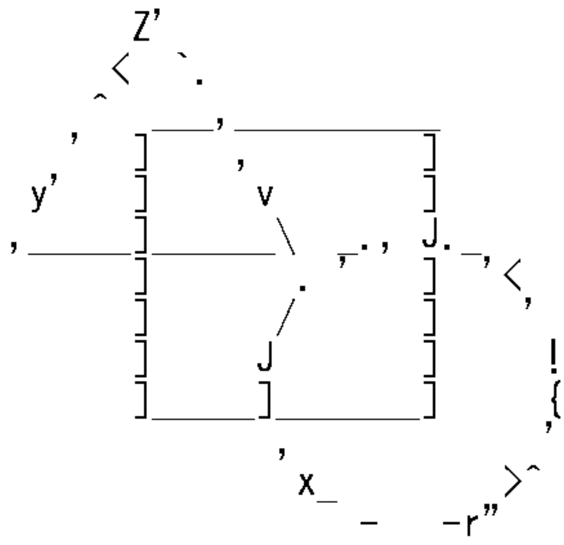



Figure 7: *The results of template matching.*

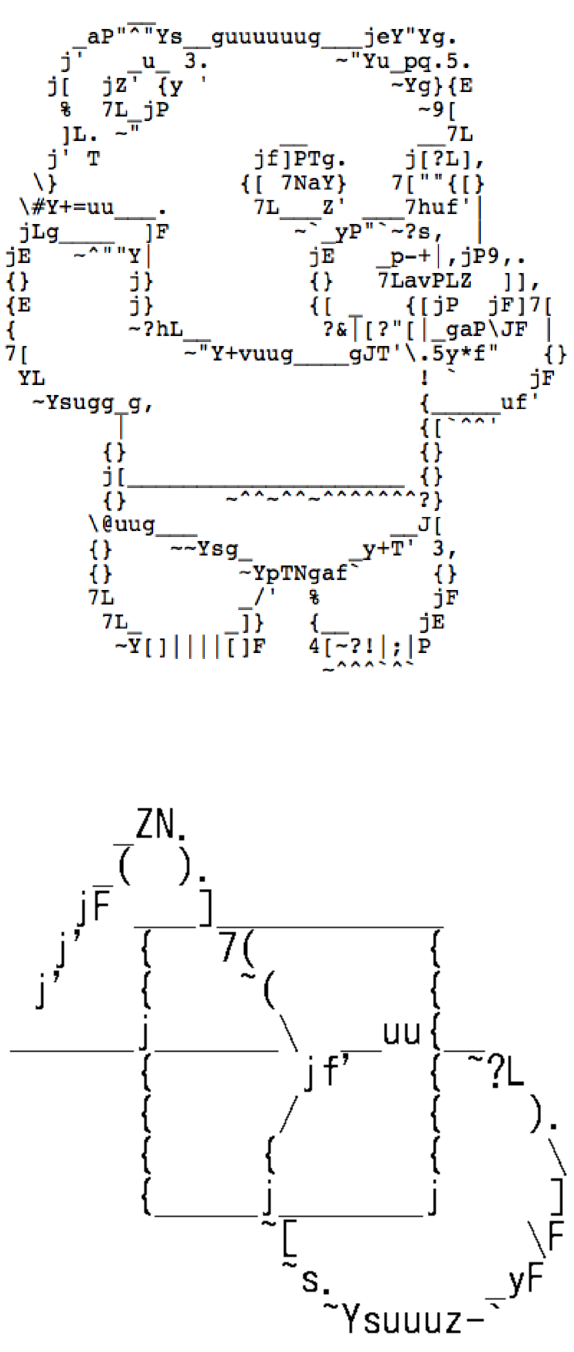


Figure 8: *The results of NCC.*

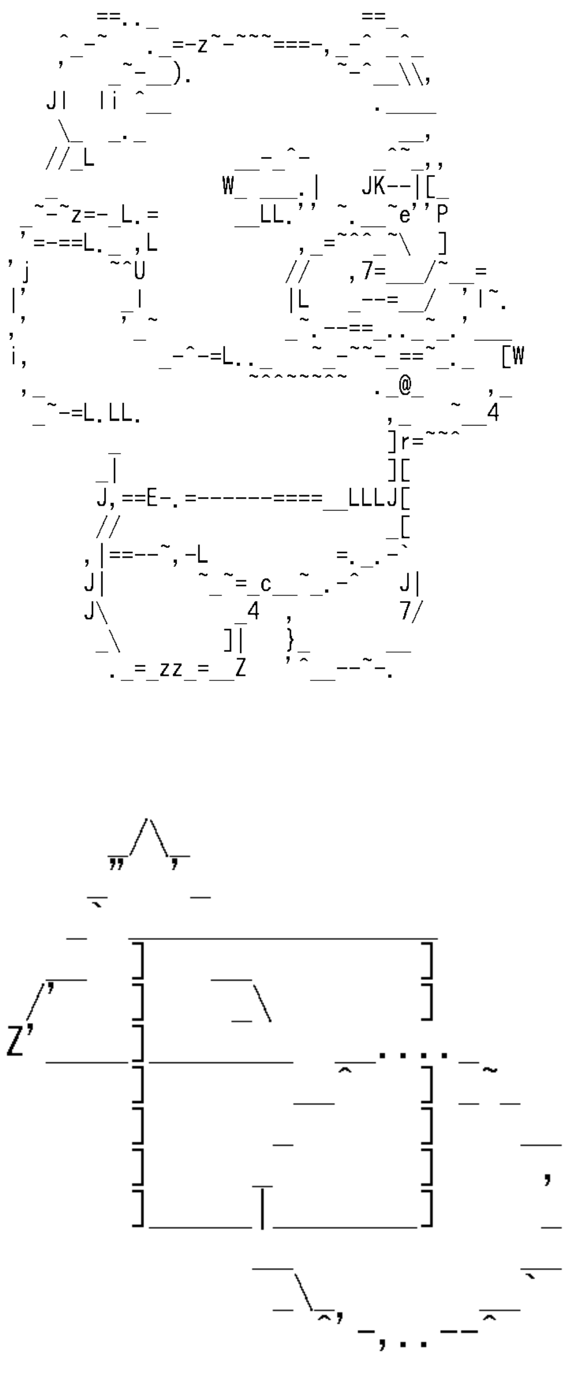


Figure 9: *The results of HOG.*

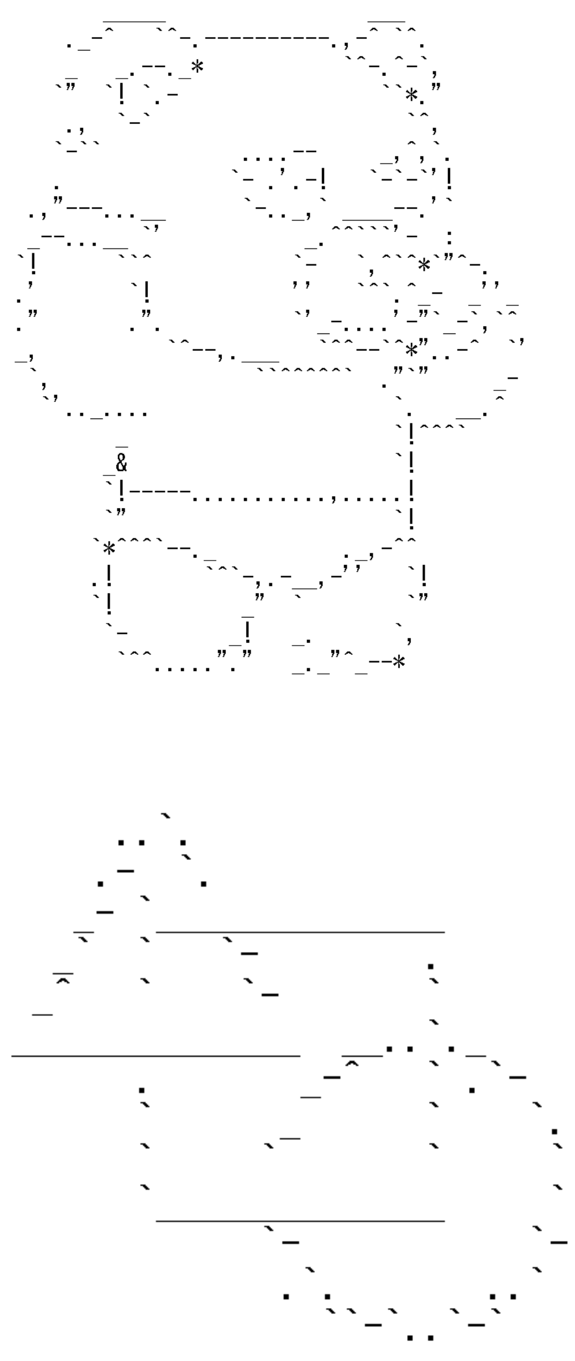


Figure 10: *The results of distance transformation.*

Our method is faster compared to the method proposed by Xu et al. [3] which takes about 10 minutes (Table 1). As the result, our system allows users to interactively create ASCII art images.

Table 1: *Computational time using the four matching metrics.*

Computational time (sec)	Bear	Shapes
Template matching	0.43	0.15
NCC	0.41	0.12
HOG	1.56	0.49
Distance Transformation	0.36	0.12

6 Conclusions and Future Work

In this paper, we have proposed an interactive system which creates structure-based ASCII art in a few seconds from images or illustrations which are drawn by users using image gradient and distance transformation. In our system, users can see the results immediately. Our system allows users to also edit the glyphs of the results by choosing from displayed candidates.

Based on experimental results, we found that the suitable matching metrics to produce visually pleasant ASCII art images depend on the type of the input images. Therefore, we need to have objective evaluations by user test. We will have users choose the best results for input images. We would like to create ASCII art which consists of proportional font. The glyphs of proportional font are of different width size. ASCII art of proportional font are often used in Japanese bulletin board system, such as 2channel.

References

[1] Jan Hubicka. AA-lib. http://aa-project.sourceforge.net/aalib/.

[2] Paul D. O'Grady and Scott T. Rickard. Automatic ascii art conversion of binary images using non-negative constraints. In *Proceedings of the Irish Signal and Systems Conference*, pages 186–191, 2008.

[3] Xuemiao Xu, Linling Zhang, and Tien-Tsin Wong. Structure-based ascii art. *ACM Transactions on Graphics (SIGGRAPH 2010 issue)*, 29(4):52:1–52:9, July 2010.

[4] Kai Briechle and Uwe D. Hanebeck. Template matching using fast normalized cross correlation. In *Proceedings of SPIE*, volume 4387, page 95, 2001.

[5] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference Computer Vision and Pattern Recognition*, pages 886 – 893, 2005.

[6] Intel Corporation and Willow Garage. OpenCV. http://opencv.willowgarage.com/.