# Structure-based ASCII Art

Xuemiao Xu* Linling Zhang† Tien-Tsin Wong‡
The Chinese University of Hong Kong

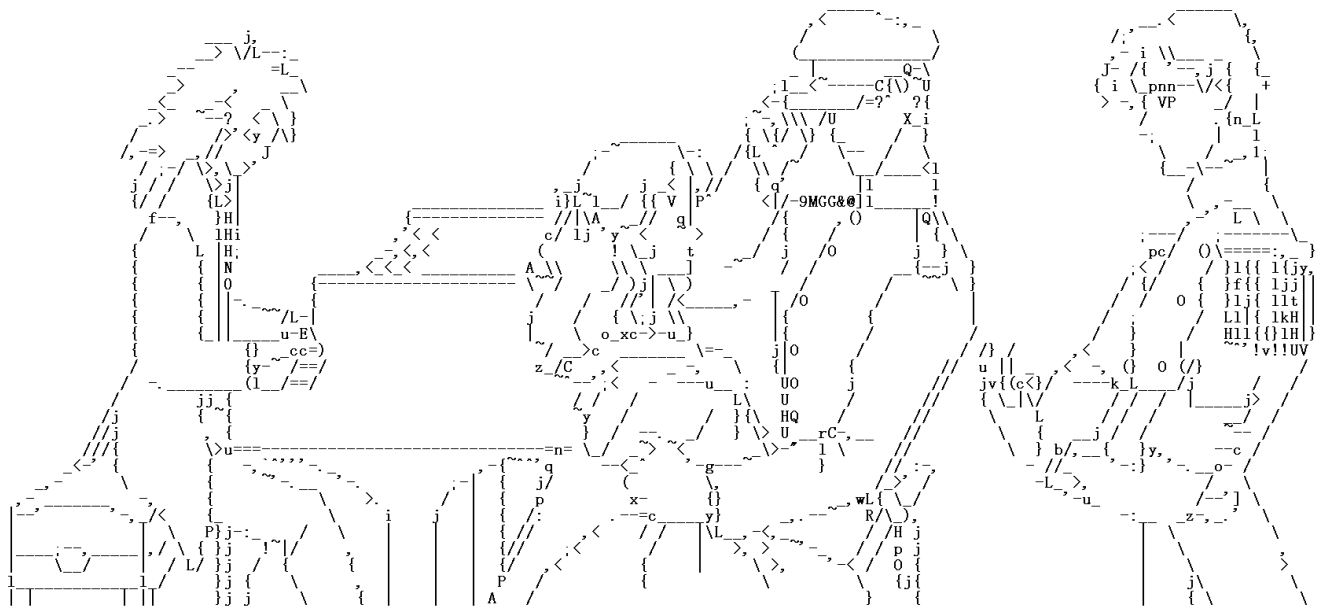


Figure 1: Structure-based ASCII art generated by our method (the input is "banquet" of Figure 18). Characters were chosen from the set of 95 printable ASCII characters.

## Abstract

The wide availability and popularity of text-based communication channels encourage the usage of ASCII art in representing images. Existing tone-based ASCII art generation methods lead to halftone-like results and require high text resolution for display, as higher text resolution offers more tone variety. This paper presents a novel method to generate *structure-based* ASCII art that is currently mostly created by hand. It approximates the major line structure of the reference image content with the shape of characters. Representing the unlimited image content with the extremely limited shapes and restrictive placement of characters makes this problem challenging. Most existing shape similarity metrics either fail to address the misalignment in real-world scenarios, or are unable to account for the differences in position, orientation and scaling. Our key contribution is a novel *alignment-insensitive shape similarity (AISS) metric* that tolerates misalignment of shapes while accounting for the differences in position, orientation and scaling. Together with the constrained deformation approach, we formulate the ASCII art generation as an optimization that minimizes *shape dissimilarity* and *deformation*. Convincing results and user study are shown to demonstrate its effectiveness.

**Keywords:** ASCII art, shape similarity

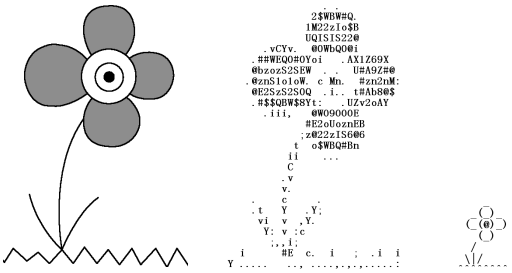
*e-mail: xmxu@cse.cuhk.edu.hk
†e-mail: llzhang@cse.cuhk.edu.hk
‡e-mail: ttwong@cse.cuhk.edu.hk

## 1 Introduction

ASCII art is a technique of composing pictures with printable text characters [Wikipedia 2009]. It stemmed from the inability of graphical presentation on early computers. Hence text characters are used in place of graphics. Even with the wide availability of digital images and graphics nowadays, ASCII art remains popular due to the enormous growth of text-based communication channels over the Internet and mobile communication networks, such as instant messenger systems, Usenet news, discussion forums, email and short message services (SMS). In addition, ASCII art has already evolved into a popular art form in cyberspace.

ASCII art can be roughly divided into two major styles, tone-based and structure-based. While tone-based ASCII art maintains the intensity distribution of the reference image (Figure 2(b)), structure-based ASCII art captures the major structure of the image content (Figure 2(c)). In general, tone-based ASCII art requires a much higher text resolution to represent the same content than the

**ACM Reference Format**
Xu, X., Zhang, L., Wong, T. 2010. Structure-based ASCII Art. *ACM Trans. Graph. 29*, 4, Article 52 (July 2010), 9 pages. DOI = 10.1145/1778765.1778789 http://doi.acm.org/10.1145/1778765.1778789.



(a) (b) (c)

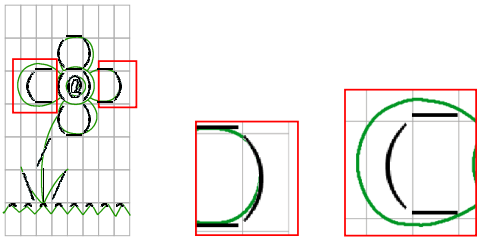
Figure 2: ASCII art. (a) A reference image. (b) Tone-based ASCII art generated by the program PicText, requiring the text resolution 30×29 in order to depict the content, though not very clearly. (c) Structure-based ASCII art manually designed by an artist, with a significant lower text resolution of 8×7.

structure-based one, as the high text resolution is required for producing sufficient tone variety. On the other hand, structure-based ASCII art utilizes the shape of characters to approximate the image structure (Figure 2(c)), without mechanically following the pixel values. To the extreme, smileys, such as `:)` and `:(`, are the simplest examples of structure-based ASCII art.

Existing computational methods can only handle tone-based ASCII art, as its generation can be regarded as a dithering problem with characters [Ulichney ]. O'Grady and Rickard [2008] improved such dithering process by reducing the mismatches bewteen character pixels and the reference image pixels. Nevertheless, high text resolution is still required for a clear depiction. Note that ASCII art gradually loses its stylishness (and approaches to standard halftone images) as its text resolution increases. In addition, as the text screens of mobile devices are limited, the character-saving structure-based ASCII art is more stylish and practical for commercial usage such as text-based advertisement. However, satisfactory structure-based ASCII art is mostly created by hand. The major challenge is the inability to depict the unlimited image content with the limited character shapes and the restrictive placement of characters over the character grid.

To increase the chance of matching appropriate characters, artists tolerate the misalignment between the characters and the reference image structure (Figure 3(b)), and even intelligently deform the reference image (Figure 3(c)). In fact, shape matching in ASCII art application is a general pattern recognition problem. In real-world applications, such as optical character recognition (OCR) and ASCII art, we need a metric to *tolerate misalignment* and also *account for the differences in transformation* (translation, orientation and scaling). For instance, in recognizing the characters "○" and "●" during the OCR, both scaling and translation count; while in recognizing characters "6" and "9", the orientation counts. Unfortunately, existing shape similarity metrics are either alignment-sensitive [Wang et al. 2004] or transformation-invariant [Mori et al. 2005; Belongie et al. 2002; Arkin et al. 1991], and hence not applicable.

In this paper, we propose a novel method to generate structure-based ASCII art to capture the major structure of the reference image. Inspired by the two matching strategies employed by ASCII artists, our method matches characters based on a novel *alignment-insensitive shape similarity metric* and allows a constrained deformation of the reference image to increase the chance of character matching. The proposed similarity metric tolerates the misalignment while it accounts for the differences in transformation. Given an input and a target text resolution, we formulate the ASCII art generation as an optimization by minimizing the shape dissimilarity and deformation. We demonstrate its effectiveness by several convincing examples and a user study. Figure 1 shows the result automatically obtained by our method.



(a) (b) (c)

Figure 3: (a) By inspecting the overlapping image between the edge map of the reference image (Figure 2(a)) and the structured-based ASCII art (Figure 2(c)), one can identify the two matching strategies employed by ASCII artists: (b) misalignment is tolerated; (c) the reference image is deformed to increase the chance of matching.

## 2 Related Work

As a culture in the cyberspace, the best references of ASCII art can be found online. There is collaboratively prepared frequently asked questions (FAQ) for Usenet newsgroup `alt.ascii-art` [CJRandall 2003], which keeps track of the update information and resources related to ASCII art. Other sources of reference are online tutorials written by individual enthusiasts [Wakenshaw 2000; Crawford 1994; Au 1995]. To produce ASCII art, one can type it using a standard text editor. It is not as intuitive as painting, however. Enthusiasts developed interactive *painting* software [Davis 1986; Gebhard 2009] to allow users to directly *paint* the characters via a painting metaphor.

Besides the interactive tools, there are attempts to automatically convert images into ASCII art [Klose and McIntosh 2000; DeFusco 2007; O'Grady and Rickard 2008]. However, they can only generate tone-based ASCII art, as it can be regarded as a dithering process. The major academic study is in the area of halftoning [Ulichney ; Bayer 1973; Floyd and Steinberg 1974]. O'Grady and Rickard [2008] tailor-made a method for tone-based ASCII art by minimizing the difference between the characters and the reference image in a pixel-by-pixel manner. However, all these methods cannot be extended to generate structure-based ASCII art due to their inability to allow misalignment and deformation. In this paper, we focus on the generation of structure-based ASCII art as it depicts a clearer picture within a smaller text space. Its generation can no longer be regarded as a dithering process. Instead, the shape similarity plays a major role in its generation. 3D collage [Gal et al. 2007] relies on shape matching to aggregate smaller objects to form a large compound one. While transformation invariance is needed during collaging, our character matching must be transformation-aware and with restrictive placement.

## 3 Overview

An overview of our structure-based ASCII art generation is shown in Figure 4. The basic input is a vector graphics containing only polylines. A raster image can be converted to vector via vectorization. As the limited shapes and restrictive placement of text characters may not be able to represent unlimited image content, ASCII artists slightly deform the input to increase the chance of character matching. So we mimic such deformation during optimization by iteratively adjusting the vertex positions of the input polylines. Given the vector-based line art, we rasterize it and divide the raster image into grid cells. Each cell is then best-matched with a character based on the proposed alignment-insensitive shape similarity metric (Section 4). This completes one iteration of optimization, and the objective value, which composes of the *deformation of the vectorized picture* (Section 5) and the *dissimilarity* between the

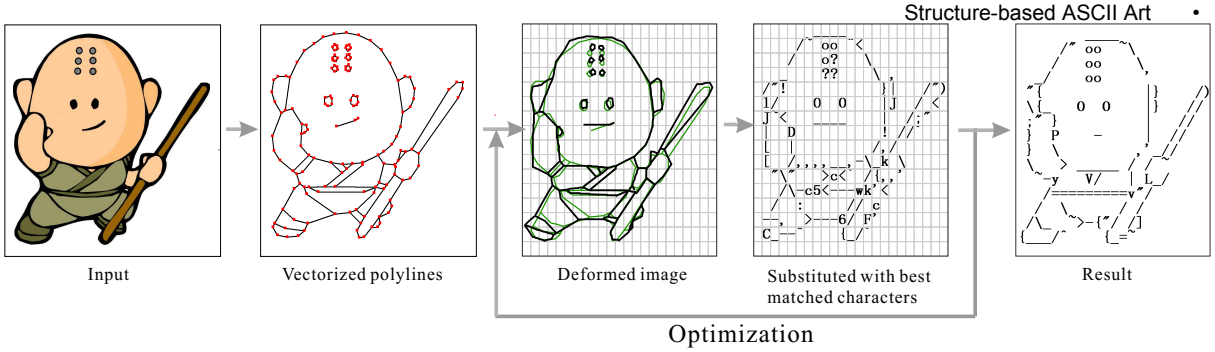



Figure 4: The overview of our framework.

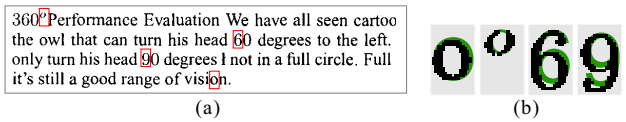



Figure 5: Real-world applications, like OCR and ASCII art, require a similarity metric to account for scaling, translation and orientation, as well as tolerate misalignment. (a) A scanned image for OCR input. (b) Misalignment with ideal characters (in green) exists.

characters and the deformed picture, can be computed. In the next iteration, we adjust the vertex positions of the vector-based line art with a simulated annealing strategy (detailed in Section 5). Since the line art is changed, the above rasterization-and-AISS-matching process is repeated to obtain a new set of best-matched characters. Such deformation-and-matching process continues until the objective value is minimized.

Before the optimization, we need to prepare the input and the characters. Since character fonts may have varying thicknesses and widths, we simplify the problem by ignoring font thickness (via centerline extraction) and handling only fixed-width character fonts. We further vectorize the characters and represent them with polylines. In order to focus only on the shapes during matching, both the input polylines and the characters are rasterized with the same line thickness (one pixel-width in our system). Note that the characters are only rasterized once as they can be repeatedly used. Before each optimization step, the input polylines are rasterized according to the target text resolution, $R_w \times R_h$, where $R_w$ and $R_h$ are the maximum number of characters along the horizontal and vertical directions respectively. As the aspect ratio of our characters, $\alpha = T_h/T_w$, is fixed, the text resolution can be solely determined by a single variable $R_w$, as $R_h = \lceil H/(\alpha\lceil W/R_w\rceil)\rceil$, where $T_w$ and $T_h$ are the width and height of a rasterized character image in the unit of pixels respectively. $W$ and $H$ are the width and height of the input image. Hence, the input polylines are scaled and rasterized to a domain of $T_wR_w \times T_hR_h$. Furthermore, since the vector-based input is scalable ($W$ and $H$ can be scaled up or down), users may opt for allowing the system to determine the optimal text resolution ($R_w \times R_h$) by choosing the minimized objective values among results of multiple resolutions, as our objective function is normalized to the text resolution.

## 4 Alignment-Insensitive Shape Similarity

The key to best-match the content in a grid cell with a character is the shape similarity metric. It should tolerate misalignment and, simultaneously, account for the differences in transformation such as, position, orientation and scaling. Existing shape similarity metrics can be roughly classified into two extreme categories, alignment-sensitive metrics and transformation-invariant metrics.

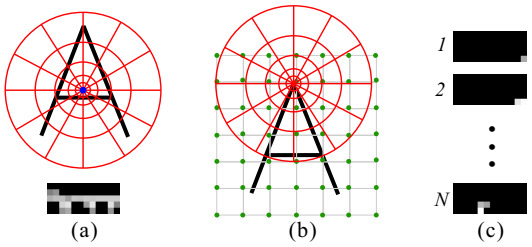



Figure 6: Alignment-insensitive shape similarity. (a) A log-polar diagram to quantify the letter "A" with the corresponding histogram underneath. Its row and column correspond to the angular and radial dimensions of the log-polar diagram respectively. (b) $N$ points are regularly sampled in a grid layout, each with a log-polar diagram. (c) The corresponding log-polar histograms.

Peak signal-to-noise ratio (PSNR) or mean-squared error (MSE), and the well-known structural similarity index (SSIM) [Wang et al. 2004] belong to the former category. Their similarity values drop significantly when two equal images are slightly misaligned during the comparison. On the other hand, the transformation-invariant metrics are designed to be invariant to translation, orientation and scaling. These metrics include shape context descriptor [Mori et al. 2005; Belongie et al. 2002], Fourier descriptor [Zahn and Roskies 1972], skeleton-based shape matching [Sundar et al. 2003; Goh 2008; Torsello and Hancock 2004], curvature-based shape matching [Cohen et al. 1992; Milios 1989], and polygonal shape matching [Arkin et al. 1991]. In our case, the transformation matters. Hence, no existing work is suitable for our application.

In fact, the above metric requirement is not only dedicated to our application, but applicable for real-world applications of pattern recognition and image analysis, such as OCR. For example, Figure 5(a) shows a scanned image ready for OCR. The characters "°", "○", "6" and "9" are magnified in Figure 5(b) for better visualization. It is not surprising that the scanned character images (in black) may be slightly misaligned to the ideal characters (in green) no matter how perfect the global registration is. Hence, an alignment insensitive shape similarity metric is essential. Besides the misalignment, the transformation difference has to be accounted for in OCR as well. Characters "○" and "°" have the similar shapes, but are different in position and scaling. Characters "9" and "6" also share the same shape but with a difference in orientation. In other words, the shape information alone is not sufficient for recognition, since position, orientation and scaling have their own special meanings. Therefore, the desired metric must also account for position, orientation, scaling, as well as the shape information.

**Misalignment Tolerance** Misalignment is, in essence, a small-scale transformation. To tolerate misalignment, a histogram of a log-polar diagram [Mori et al. 2005] is used as the basic building

block of our shape descriptor (Figure 6(a)). This log-polar histogram measures the shape feature in a local neighborhood, covered by a log-polar window. Its bins uniformly partition the local neighborhood in log-polar space. For each bin, the grayness of the shape is accumulated and used as one component in the histogram. As the bins are uniform in log-polar space, the histogram is more sensitive to the positions of nearby points than to those farther away. Moreover, since only the sum of pixels within the same bin is relevant, it is inherently insensitive to small shape perturbations, which leads to its misalignment tolerance nature. In other words, the degree of misalignment tolerance is implicitly defined in the log-polar diagram. During the pixel summation, black pixel has a grayness of 1 while the white one is 0. The bin value $h(k)$ of the $k$-th bin is computed as $h(k) = \sum_{(q-p)\in \mathrm{bin}(k)} \mathbf{I}(q)$, where $q$ is the position of the current pixel; $(q - p)$ is the relative position to the center of the log-polar window, $p$; $\mathbf{I}(q)$ returns the grayness at position $q$. The lower sub-image in Figure 6(a) visualizes the feature vector $h$ with respect to $p$ (the blue dot).

**Transformation Awareness** Unlike the original transformation-invariance scheme in [Mori et al. 2005], we propose a novel sampling layout of log-polar diagrams in order to account for the transformation difference. The log-polar histogram can natively account for orientation. The bin values change as the content rotates. To account for scaling, all log-polar histograms share the same scale. To account for translation (or position), $N$ points are regularly sampled over the image in a grid layout (Figure 6(b)). Both the reference image in a cell and the character image are sampled with the same sampling pattern. For each sample point, a log-polar histogram is measured. The feature vectors (histograms) of the sample points are then concatenated to describe the shape, as shown in Figure 6(c). The shape similarity between two shapes, $S$ and $S'$, is measured by comparing their feature vectors in a point-by-point basis, given by

$$D_{\mathrm{AISS}}(S, S') = \frac{1}{M} \sum_{i \in N} ||\mathbf{h}_i - \mathbf{h}'_i||, \qquad (1)$$

where $\mathbf{h}_i$ ($\mathbf{h}'_i$) is the feature vector of the $i$-th sample point on $S$ ($S'$); $M = (n+n')$ is the normalization factor and $n$ ($n'$) is the total grayness of the shape $S$ ($S'$). This normalization factor counteracts the influence of absolute grayness.

In all the experiments, histograms were empirically constructed with 5 bins along the radial axis in log space, and 12 bins along the angular axis. The radius of the coverage is selected to be about half of the shorter side of a character. The number of sample points, $N$, equals $(T_w/2) \times (T_h/2)$. To suppress aliasing due to the discrete nature of bins, the image is filtered by a Gaussian kernel of size 7×7 before measuring the shape feature.

**Comparison to Existing Metrics** We evaluate the metric by comparing it to three commonly used metrics, including the classical shape context (a translation- and scale- invariant metric), SSIM (an alignment-sensitive, structure similarity metric), and RMSE (root mean squared error) after blurring. For the last metric, RMSE is measured after blurring the compared images by a Gaussian kernel of 7×7, as one may argue that our metric is similar to RMSE after blurring the images.

The effectiveness of our metric is demonstrated in Figure 7, in which we query four different shapes (the first column). For each metric, the best-matched character is determined from a set of 95 printable ASCII characters. From the matching results, shape context over-emphasizes the shape and ignores the position (as demonstrated by queries 2 to 4). On the other hand, the alignment-sensitive nature of SSIM and RMSE drives them to maximize the overlapping area between the query image and the character, while

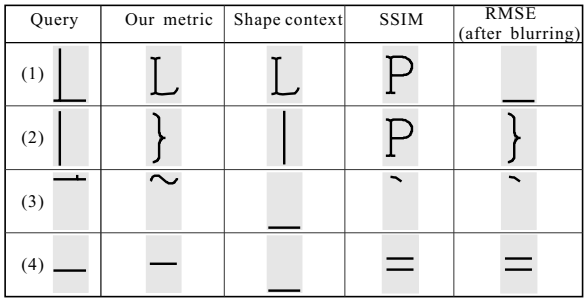
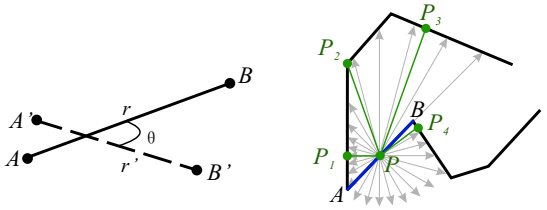



Figure 7: Comparison of four shape similarity metrics. From left to right: our metric, shape context, SSIM, and RMSE-after-blurring.




(a) Local deformation (b) Accessibility

Figure 8: Deformation metric

paying less attention to the shape (demonstrated by queries 1 and 3). In contrast, our method strives for a balance between shape and position in all results. The query of a long center-aligned horizontal line (query 4) demonstrates the advantage of our metric. Shape context maximizes shape similarity, ignores large displacement, and chooses the longer underscore character "_" to match the long line. SSIM and RMSE match the shape with an equal sign "=" because its lower line overlaps with the query image. Our method pays attention to the shape (a single line), tolerates a slight misalignment, and chooses a shorter hyphen "–" as the best match.

## 5 Optimization

**Deformation Metric** To raise the chance of matching characters, ASCII artists intelligently deform the reference image. We mimic such deformation during our optimization. We deform the reference image by adjusting the vertex positions of the vectorized polylines. However, unconstrained deformation may destroy the global structure of the input. We designed a metric to quantify and minimize the deformation values during the optimization process. This consists of two terms, *local deformation constraint* and *accessibility constraint*.

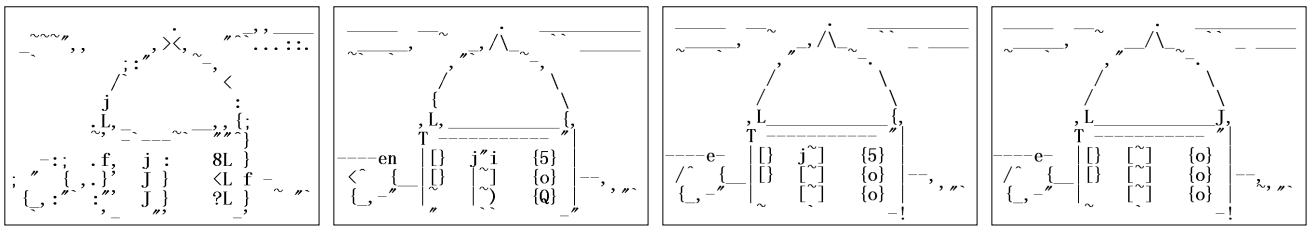
*Local Deformation Constraint* The first term measures the local deformation of a line segment, in terms of orientation and scaling. Consider the original line segment $AB$ as deformed to $A'B'$ in Figure 8(a). As we allow global translation during the deformation, the local deformation of line segment $AB$ is measured in a relative sense, as follows,

$$D_{\mathrm{local}}(AB) = \max\left\{V_\theta(AB), V_r(AB)\right\}, \qquad (2)$$

where $V_\theta(AB) = \exp(\lambda_1 \theta)$, and

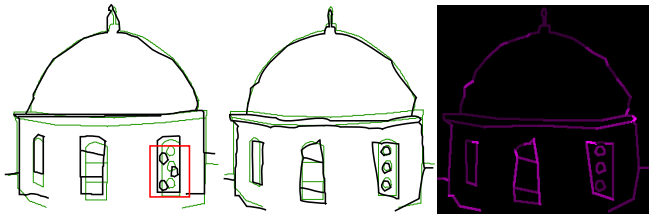
$$V_r(AB) = \max\left\{\exp(\lambda_2 |r' - r|), \exp\left(\frac{\lambda_3 \max\{r, r'\}}{\min\{r, r'\}}\right)\right\},$$

$\theta \in [0, \pi]$ is the angle between the original and the deformed line segments. $r$ and $r'$ denote the lengths of the original and deformed



(a) Iteration 0 $E$ = 695.76 (b) Iteration 60 $E$ = 425.33 (c) Iteration 120 $E$ = 375.32 (d) Iteration 155 $E$ = 365.76

Figure 10: Intermediate results during optimization. The input is Figure 18(s3).



(a) (b) (c)

Figure 9: The green and black lines indicate the original and deformed polylines respectively. The input is Figure 18(s3). (a) With the local deformation constraint alone, the drift of circular windows cannot be avoided. (b) With local and accessibility constraints, the drift can be avoided. (c) Visualization of the deformation value of each line segment in (b). For visualization purpose, the deformation values are non-linearly mapped.

line segments. Parameters $\lambda_1$, $\lambda_2$, and $\lambda_3$ are the weights, and empirically set to values of $8/\pi$, $2/\min\{T_w, T_h\}$, and 0.5, respectively, in all the experiments. When there is no local deformation, $D_{\text{local}} = 1$.

*Accessibility Constraint* The local deformation constraint alone can only prevent the over-deformation in the local scale. It cannot avoid the over-deformation in a global scale, as demonstrated in Figure 9(a). Three circular windows drift away from their original locations and destroy the layout, even though each of them is not over-deformed in a local sense. To constrain the deformation in a global scale, we propose a 2D accessibility constraint, inspired by the surface exposure [Hsu and Wong 1995] and 3D accessibility [Miller 1994]. This maintains the relative orientation and position between the current line segment and its surrounding line segments.

To compute the accessibility of the current line segment, say $AB$ in Figure 8(b), multiple rays are shot from the midpoint, $P$, of $AB$ towards the surrounding circle in order to determine the closest surrounding line segments. For each intersected line segment, the nearest point, $P_i$, is determined, forming an imaginary line segment $PP_i$. The accessibility is then computed as

$$D_{\text{access}}(AB) = \sum_{i=1}^{n_l} w_i D_{\text{local}}(PP_i), \tag{3}$$

where $n_l$ is the total number of intersecting line segments to $P$. $D_{\text{local}}(PP_i)$ is defined in Equation 2; $w_i$ is the weight, computed as the normalized distance $w_i = |PP_i|/(\sum_{i=1}^{n_l} |PP_i|)$. Its value is higher when the corresponding line segment is closer to $P$. Hence, the overall metric of controlling the deformation is,

$$D_{\text{deform}}(AB) = \max\{D_{\text{local}}(AB), D_{\text{access}}(AB)\}, \tag{4}$$

where $D_{\text{deform}} = 1$ when there is no deformation. Figure 9(c) visualizes $D_{\text{deform}}$ of the deformed image (Figure 9(b)) by color-coding each line segment with lighter value indicating higher deformation, and vice versa. As the objective function is computed on the basis of a character cell, the deformation value of a character cell $j$, $D^j_{\text{deform}}$, is computed. All line segments intersecting the current cell $j$ are identified, as denoted by the set $\{\mathcal{L}_j\}$. $l_i$ is the length of the $i$-th line segment $L_i$ (partial or whole) in $\{\mathcal{L}_j\}$ occupied by cell $j$. Then, the deformation value of cell $j$ is then computed as the weighted average of deformation values of involved line segments,

$$D^j_{\text{deform}} = \sum_{i \in \{\mathcal{L}_j\}} \tilde{l}_i D_{\text{deform}}(L_i), \quad \text{where } \tilde{l}_i = \frac{l_i}{\sum_{i \in \{\mathcal{L}_j\}} l_i}. \tag{5}$$

**Objective Function** With the shape similarity and deformation metrics, the overall objective function can be defined. Given a particular text resolution, our optimization goal is to minimize the energy $E$,

$$E = \frac{1}{K} \sum_{j=1}^{m} D^j_{\text{AISS}} \cdot D^j_{\text{deform}}, \tag{6}$$

where $m$ is the total number of character cells, and $K$ is the number of non-empty cells, and is used as the normalization factor. $D^j_{\text{AISS}}$ is the dissimilarity between the $j$-th cell's content and its best-matched character, as defined in Equation 1. The term $D^j_{\text{deform}}$ is the deformation value of the $j$-th cell. When there is no deformation, $D^j_{\text{deform}} = 1$; hence $E$ is purely dependent on $D^j_{\text{AISS}}$. Note that the energy values of different text resolutions are directly comparable, as our energy function is normalized. The lower row of Figure 12 demonstrates such comparability by showing our results in three text resolutions along with their energies. The middle one (28×21) with the smallest energy corresponds to the most pleasant result, while the visually poor result on the left has a relatively larger energy.

We employ a simulated annealing strategy during the discrete optimization. In each iteration, we randomly select one vertex, and randomly displace its position with a distance of at most $d$. Here, $d$ is the length of the longer side of the character image. Then, all affected grid cells due to this displacement are identified and best-matched with the character set again. If the recomputed $E$ is reduced, the displacement is accepted. Otherwise, a transition probability $P_r = \exp(-\delta/t)$ is used to make the decision, where $\delta$ is the energy difference between two iterations; $t = 0.2 t_a c^{0.997}$ is the temperature; $c$ is the iteration index; $t_a$ is the initial average matching error of all grid cells. If $P_r$ is smaller than a random number in $[0, 1]$, this displacement is accepted; otherwise, it is rejected. The optimization is terminated whenever $E$ is not reduced for $c_o$ consecutive iterations, where $c_o = 5000$ in our implementation.

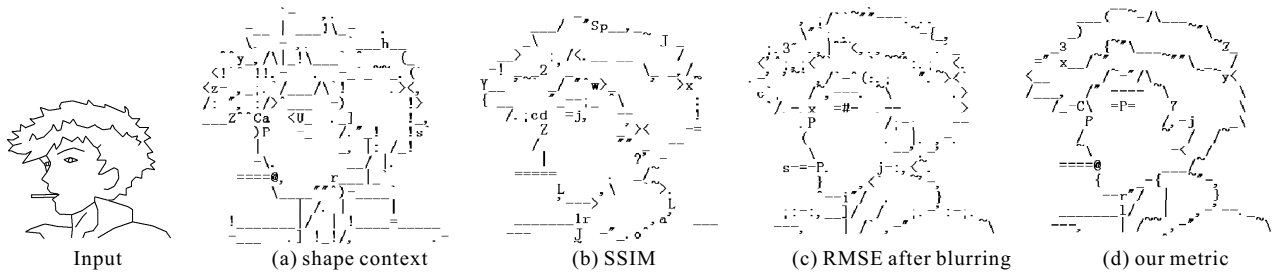



Figure 11: Comparison of ASCII art using different shape similarity metrics.

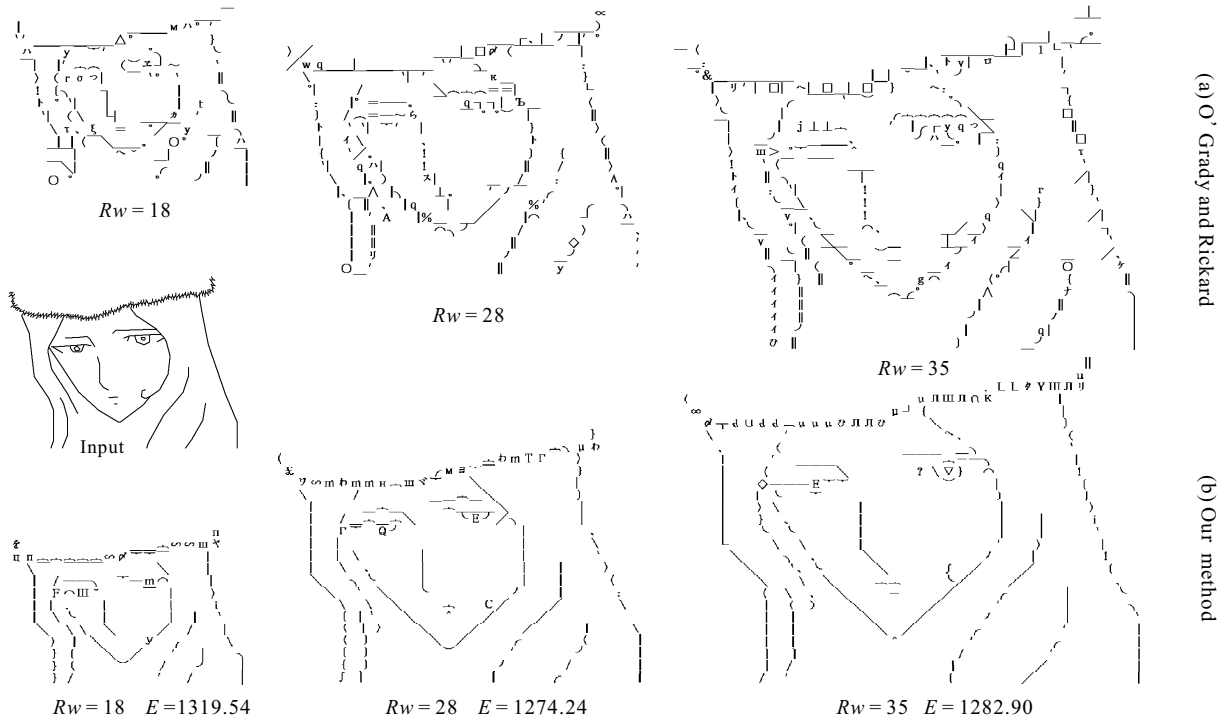



Figure 12: Our method vs. the method of O'Grady and Rickard. $R_w$ is the width of the text resolution and $E$ is the optimized energy.

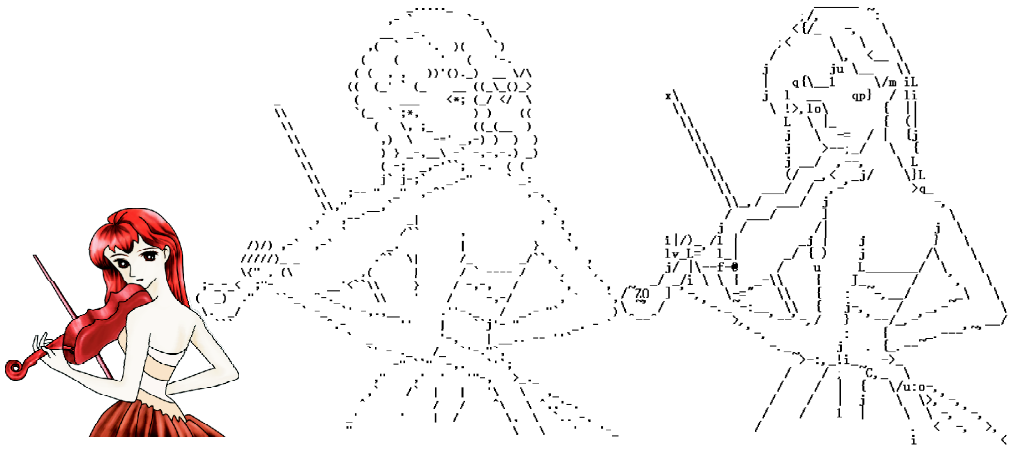
Figure 10 shows the intermediate results along with their energies. As the energy reduces, the visual quality of ASCII art improves accordingly. An animated sequence for better visualization of such optimization is included in the auxiliary material.

## 6 Results and Discussions

To validate our method, we conducted multiple experiments over a rich variety of inputs. The setting used for generating all our examples in this paper and their running times are listed in Table 2. Our method works with any font database of fixed character width. This paper shows results of matching characters from ASCII code (95 printable characters) and Shift-JIS code (475 characters only). Figures 14 to 17 show our results. The corresponding inputs can be found in Figure 18. Complete comparisons and results can be found in the auxiliary material.

**Metrics Comparison** In Section 4, we have compared different shape similarity metrics for matching a *single* character. One may argue the visual importance of the proposed metric in generating the entire ASCII art which may contain hundreds of characters. To validate its importance, we compare the ASCII art results (Figure 11) generated by substituting the character matching metric in our framework with different shape similarity metrics, including shape context, SSIM, RMSE after blurring and our metric. Hence, the same deformation mechanism is employed in generating all results. The result of shape context (Figure 11(a)) is most unrecognizable due to the structure discontinuity caused by the neglect of position. SSIM and RMSE preserve better structure as they place a high priority on position. Their alignment-sensitive nature, however, leads to the loss of fine details. Among all results, our metric generates the best approximation to the input, with the best preservation of structure and fine details. The comparison demonstrates the importance of transformation awareness and misalignment tolerance in preserving structure continuity and fine details.

**Comparison to Existing Work** Figure 2 already demonstrates the inferiority of the more naïve halftoning approach in representing clear structure. The only alternative work that was tailormade for generating ASCII art is by O'Grady and Rickard [2008]. We therefore compare our results to those generated by their method in Figure 12(a). Due to its halftone nature, their method fails to produce satisfactory (in terms of structure preservation) results for



(a) Input (b) by artist (c) by our method

Figure 13: Comparison of ASCII art between an artist's and our method.

| | Text resolut. | Character set | Running time | Time by artists |
|---|---|---|---|---|
| Fig. 1 | 138X36 | ASCII | 15mins | 5 hrs |
| Fig. 11 | 26X16 | ASCII | 4mins | - |
| Fig. 12b | 18X13 | Shift-JIS | 6mins | - |
| Fig. 12b | 28X21 | Shift-JIS | 8mins | - |
| Fig. 12b | 35X28 | Shift-JIS | 8mins | - |
| Fig. 13 | 52X34 | ASCII | 7mins | 2 hrs |
| Fig. 14 | 66X61 | ASCII | 12mins | - |
| Fig. 15 | 31X20 | Shift-JIS | 15mins | - |
| Fig. 16 | 35X27 | Shift-JIS | 11mins | - |
| Fig. 17a | 24X37 | Shift-JIS | 9mins | - |
| Fig. 17b | 24X38 | Shift-JIS | 9mins | - |
| Fig. 17c | 25X36 | Shift-JIS | 9mins | - |

Table 2: Timing statistics.

all three text resolutions (from 18×13 to 35×28). All fine details are lost in their results.

**User Study** To conduct a user study, artists were invited to manually design ASCII art pieces for 3 test images. They were free to choose the desired text resolution for their pieces, but the character set was restricted to ASCII code. We then use our method and the method by O'Grady and Rickard to generate ASCII art results with the same text resolutions. Then, we invited 30 participants for the user study. The source image and three ASCII art results were shown side-by-side to the participants. Figure 13 shows our result as well as the artist piece from one of the 3 test sets. The complete set of comparison can be found in the auxiliary material. Each participant graded the ASCII art using 2 scores out of a 9-point scale ([1-9] with 9 being the best). The first score was to grade the similarity of the ASCII art pieces with respect to the input. The second was to grade the clarity of content presented in the ASCII art without referring to the input. Therefore, there were 18 data samples for analysis from each of the 30 participants. Altogether 540 data samples can be used for analysis.

From the statistics in Table 1, the results by O'Grady and Rickard are far from satisfactory in terms of both clarity and similarity. Our method is comparable to (but slightly poorer than) the artist production in terms of clarity. In terms of similarity, however, our method produced better results than the artist's production. Such a phenomenon can be explained by that fact that artists can intelligently (creatively) modify or even drop parts of content in order to facilitate the ASCII approximation (e.g. hairstyle of the girl in Figure 13(b)). In some cases, they even change the aspect ratio of the input to facilitate character matching. On the other hand, our method respects the input aspect ratio and content.

**Animated ASCII Art** Figure 17 shows the ASCII art results of converting an animation to a sequence of ASCII art pieces. Although each frame is converted independently without explicit maintenance of temporal coherence, the generated ASCII art sequence is quite satisfactory. Readers are referred to the auxiliary material for a side-by-side comparison between the original frames and our generated ASCII art, in an animated fashion.

**Timing Performance** The proposed system was implemented on a PC with 2GHz CPU, 8 GB system memory, and an nVidia Geforce GTX 280 GPU with 1G video memory. Table 2 summarizes the timing statistics of all examples shown in this paper. The second, third, and fourth columns show the corresponding text resolution, the character set used, and the running time for generating our ASCII art. The running time increases as the complexity of the input and the number of the characters increase.

| | Methods | Mean | Standard deviation | 95% confidence interval Lower Bound | Upper Bound |
|---|---|---|---|---|---|
| Similarity | Artists | 6.86 | 1.32 | 6.60 | 7.11 |
| | Our method | 7.36 | 1.13 | 7.14 | 7.58 |
| | O'Grady and Rickard | 4.42 | 1.82 | 4.06 | 4.77 |
| Clarity | Artists | 7.18 | 1.25 | 6.94 | 7.42 |
| | Our method | 7.09 | 1.30 | 6.84 | 7.34 |
| | O'Grady and Rickard | 4.15 | 1.80 | 3.80 | 4.50 |

Table 1: User study statistics.

**Limitations** Besides the fact that traditional ASCII art only works on a fixed-width font, modern ASCII art also deals with proportional fonts, e.g. Japanese Shift-JIS. Our current method does not handle proportional placement of characters or multiple font sizes in a single ASCII art piece. Another limitation is that we currently do not consider the temporal consistency when we generate the animation of ASCII art. To achieve this, one could first establish the correspondence between the shapes of the adjacent frames. Then one could constrain the deformation along the temporal dimension to achieve temporal consistency. Since our system only accepts vector input, real photographs or other raster images must first be converted into outline images. This could be done either by naïve edge detection or a sophisticated line art generation method such as [Kang et al. 2007], followed by vectorization. This also means that our results would be affected by the quality of the vectorization. A poorly vectorized input containing messy edges would be faithfully represented by our system. One more limitation stems from the extremely limited variety of characters. Most font sets do not contain characters representing a rich variety of slopes of lines. This makes pictures such as radial patterns very hard to be faithfully represented.

## 7 Conclusion

In this paper, we present a method that mimics how ASCII artists generate structure-based ASCII art. To achieve this, we first propose a novel alignment-insensitive metric to account for position, orientation, scaling and shape. We demonstrate its effectiveness in

balancing shape and transformations, comparing it to existing metrics. This metric should also benefit other practical applications requiring pattern recognition. Besides, a constrained deformation model is designed to mimic how the artists deform the input image. The rich variety of results shown demonstrates the effectiveness of our method. Although we have shown an application of animated ASCII art, its temporal consistency is not guaranteed. In the future, it is worth investigating the possibility of generating animations of ASCII art with high temporal consistency. An extension to proportional placement of characters is also worth studying. To further control and refine the result, it would also be beneficial to allow users to interactively highlight the important structure in the input for preservation during the deformation.

## Acknowledgments

This project is supported by the Research Grants Council of the Hong Kong Special Administrative Region, under General Research Fund (CUHK417107). We would like to thank Xueting Liu for drawing some of the line art works, and ASCII artists on `newsmth.net` including Crowyue, Zeppeli, Wolfing, and Asan for creating the ASCII arts in our comparison between the results by our method and by hand. Thanks also to all reviewers for their constructive comments and guidance in shaping this paper.

## References

ARKIN, E. M., CHEW, L. P., HUTTENLOCHER, D. P., KEDEM, K., AND MITCHELL, J. S. B. 1991. An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. Pattern Anal. Mach. Intell. 13*, 3, 209–216.

AU, D., 1995. Make a start in ASCII art. http://www.ludd.luth.se/∼vk/pics/ascii/junkyard/techstuff/tutorials/Daniel_Au.html.

BAYER, B. 1973. An optimum method for two-level rendition of continuous-tone pictures. In *IEEE International Conference on Communications*, IEEE, (26–11)–(26–15).

BELONGIE, S., MALIK, J., AND PUZICHA, J. 2002. Shape matching and object recognition using shape contexts. *IEEE Tran. Pattern Analysis and Machine Intelligience 24*, 4, 509–522.

CJRANDALL, 2003. alt.ascii-art: Frequently asked questions. http://www.ascii-art.de/ascii/faq.html.

COHEN, I., AYACHE, N., AND SULGER, P. 1992. Tracking points on deformable objects using curvature information. In *ECCV '92*, Springer-Verlag, London, UK, 458–466.

CRAWFORD, R., 1994. ASCII graphics techniques v1.0. http://www.ludd.luth.se/∼vk/pics/ascii/junkyard/techstuff/tutorials/Rowan_Crawford.html.

DAVIS, I. E., 1986. theDraw. TheSoft Programming Services.

DEFUSCO, R., 2007. MosASCII. freeware.

FLOYD, R. W., AND STEINBERG, L. 1974. An adaptive algorithm for spatial grey scale. In *SID Int.Sym.Digest Tech.Papers*, 36–37.

GAL, R., SORKINE, O., POPA, T., SHEFFER, A., AND COHEN-OR, D. 2007. 3d collage: Expressive non-realistic modeling. In *In Proc. of 5th NPAR*.

GEBHARD, M., 2009. JavE. freeware.

GOH, W.-B. 2008. Strategies for shape matching using skeletons. *Comput. Vis. Image Underst. 110*, 3, 326–345.

HSU, S.-C., AND WONG, T.-T. 1995. Simulating dust accumulation. *IEEE Comput. Graph. Appl. 15*, 1, 18–22.

KANG, H., LEE, S., AND CHUI, C. K. 2007. Coherent line drawing. In *ACM Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 43–50.

KLOSE, L. A., AND MCINTOSH, F., 2000. Pictexter. AxiomX.

MILIOS, E. E. 1989. Shape matching using curvature processes. *Comput. Vision Graph. Image Process. 47*, 2, 203–226.

MILLER, G. 1994. Efficient algorithms for local and global accessibility shading. In *Proceedings of SIGGRAPH 94*, 319–326.

MORI, G., BELONGIE, S., AND MALIK, J. 2005. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 11, 1832–1837.

O'GRADY, P. D., AND RICKARD, S. T. 2008. Automatic ASCII art conversion of binary images using non-negative constraints. In *Proceedings of Signals and Systems Conference 2008 (ISSC 2008)*, 186–191.

SUNDAR, H., SILVER, D., GAGVANI, N., AND DICKINSON, S. 2003. Skeleton based shape matching and retrieval. *SMI '03*, 130.

TORSELLO, A., AND HANCOCK, E. R. 2004. A skeletal measure of 2d shape similarity. *Computer Vision and Image Understanding 95*, 1, 1–29.

ULICHNEY, R. A. MIT Press.

WAKENSHAW, H., 2000. Hayley Wakenshaw's ASCII art tutorial. http://www.ludd.luth.se/∼vk/pics/ascii/junkyard/techstuff/tutorials/Hayley_Wakenshaw.html.

WANG, Z., BOVIK, A. C., SHEIKH, H. R., MEMBER, S., SIMONCELLI, E. P., AND MEMBER, S. 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 600–612.

WIKIPEDIA, 2009. ASCII art. http://en.wikipedia.org/wiki/Ascii_art.

ZAHN, C. T., AND ROSKIES, R. Z. 1972. Fourier descriptors for plane closed curves. *IEEE Tran. Computers 21*, 3, 269–281.

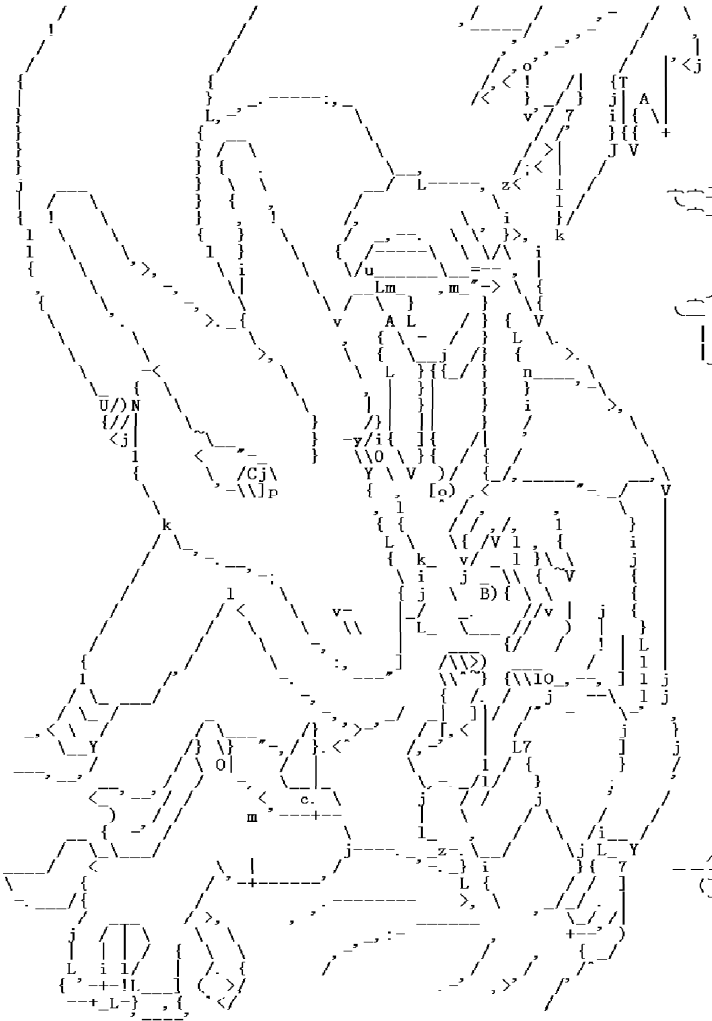


Figure 14: ASCII art of “dragon-man”

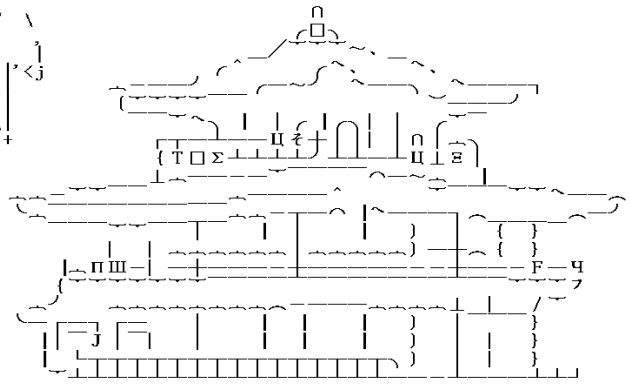


Figure 15: ASCII art of “Golden Temple”

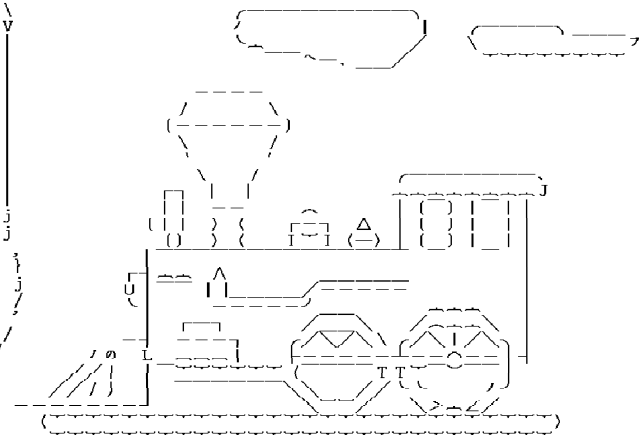
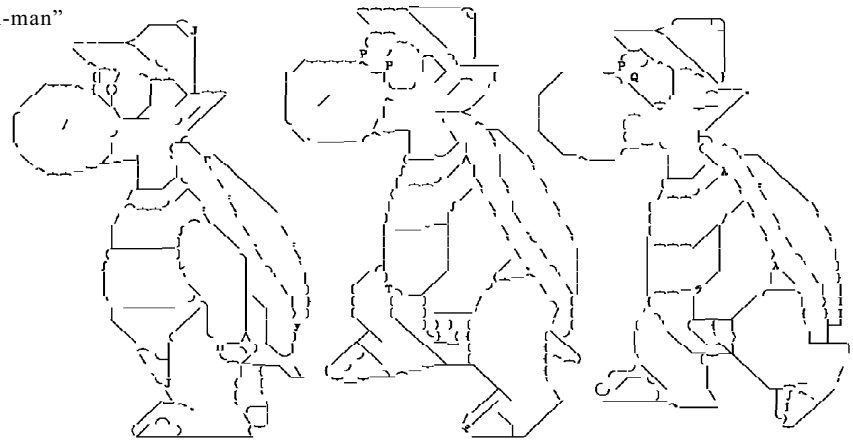


Figure 16: ASCII art of “train”

(s1) “dragon-man”

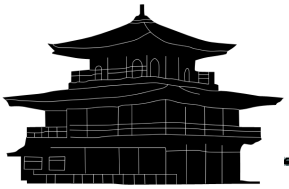


(a) Frame 1 (b) Frame 3 (c) Frame 6

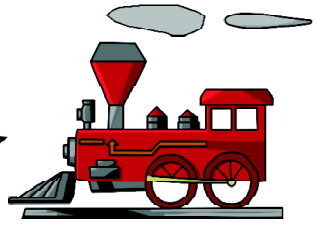
Figure 17: Animation of “toitorse”

(s2) “banquet”

(s3) “church”



(s4) “Golden Temple”



(s5) “train”

Figure 18: Inputs of examples in this paper