Vol.2010-CG-139 No.3
2010/7/16

字間調整による可読性を備えたアスキーアートの生成手法

古田陽介[†1]　三谷純[†1]　福井幸男[†1]

本稿ではグレースケールの画像とテキストファイルを入力として，テキストの濃淡によるアスキーアートを出力するシステムを提案する．既存のアスキーアートでは意図した濃淡画像を表現するために使用する文字をシステムが決定することが一般的であるが，本稿で提案するシステムでは，使用する文字列を指定することができ，その文字列を構成する文字の並び順が維持されるという特徴がある．つまり，アスキーアートを構成する文字列を，意味のある文章として読むことができ，その結果従来の濃淡画像の表現とそれに合わせたコンテキストを融合させることが可能となる．これを実現するために，平面上に配置する文字の間隔を調整することで，遠くから眺めた時の濃淡表現を行う．本研究では，提案手法を実装し，複数の画像および文字列による実験とその評価を行った．

A method for generating ASCII-Art images from a character sequence by adjusting the kerning

Yohsuke FURUTA,[†1] Jun MITANI [†1]
and Yukio FUKUI[†1]

In this paper we present a system for generating an ASCII-art image from a target grayscale image and texts which are used to represent the image. The generated ASCII-art image what we target simply is consisted with lines of the texts. However we can recognize the grayscale image by viewing it away from the printed media. The principle is simple. We adjust the density of the texts by placing characters appropriate positions so that the distribution of density in the image is represented by the lines of characters. Although the characters used in the computer generated ASCII-art are generally defined by a system automatically, users can specify the texts in our system. Further the order of the characters is maintained. Hence we can read and recognize the contexts of the texts in the ASCII-art image. We describe how to adjust the kerning of characters to generate the ASCII-art in this paper, and we show some examples.

1. はじめに

紙に印刷された，またはディスプレイに表示された文字列を遠くから眺めると，文字列を構成する各文字の画数の多寡により，色が濃く見える場所と薄く見える場所が存在する．例えばアルファベットなどと比較し，漢字などの画数の多い文字を多く含む文字列ほど濃い視覚的効果を得ることが出来る（**図 1**）．この原理を利用して，濃淡画像を文字の集合で表現したものをアスキーアートと呼ぶことがある（**図 2**）．本稿ではこれ以降,「アスキーアート」という単語を，このような文字の集合による濃淡画像の表現手法の意味で用いる．このような文字の集合よる視覚表現技法は非常に古くから存在しており，現在も Web の世界を中心にさまざまな作品を目にすることが出来る．しかしながら，そこで濃淡を表現するために用いられている文字は一般的に，元になる画像の画素の濃度にあわせてシステムが自動的に選択するものであり，アスキーアートを構成する文字の役割はいわば巨大な画素としてのものでしかなかった．そのため，完成されたアスキーアートは視覚的には意味を持っていても，文章としては意味を持たないものであった．

図 1　文字の種類の違いによる濃度の違い．左は薄く認識され，右は濃く認識される．
Fig. 1　The difference of darkness according to the types of character. Characters in right side are recognized darker than left side.

そこで我々は，画像の濃淡に対応した文字を機械的に選択するのではなく，用意した文章を構成する個々の文字の濃度に合わせその字間を変化させることで，濃淡を持ったアスキーアートを作成する手法を提案する．本手法では既存のアスキーアート生成手法と異なり，入

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba

© 2010 Information Processing Society of Japan

Vol.2010-CG-139 No.3
2010/7/16

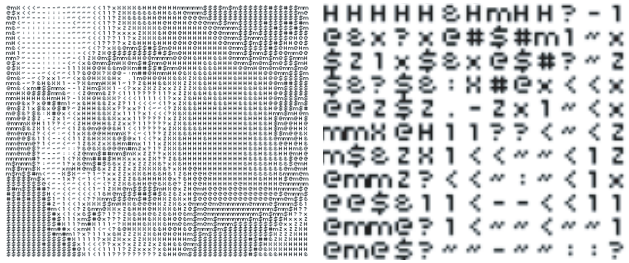


図 2 文字の濃度の違いで表現されたアスキーアートの例．右は拡大図．
Fig. 2 An example of ASCII-art in which characters express contrasting density of an image.

力された文字列の並び順を維持したまま画面や紙面上に濃淡を構築することが出来るという特徴と持つため，従来の濃淡画像表現に言語的なコンテキストを付与したり，また逆に文字列に視覚的なメッセージを付与するなどの新しい表現を行うことが可能である．また，提案システムではアスキーアートをベクトル表現のファイルフォーマットである SVG 形式で画像出力することが可能なため，画像であるにもかかわらず，ユーザはそれを選択・コピーすることで文字列としての情報を取得することが可能である．

本論文ではそのために必要となる，文字の濃度の算出方法や字間の算出方法，アスキーアートを描画する際に文章を過不足無く使い切るよう元画像を拡大縮小する手法について述べる．

2. 関 連 研 究

文字を一つのドットと見なし，文字を構成する画素の多寡により濃淡を表現するという手法は古くから存在しており，1938 年にはタイプライターを用いたものが Popular Science Magazine に紹介されている（**図 3**）[1]．この手法は計算機システムでの実装が容易であり，現在ではさまざまなプログラムが Web 上で公開されている．例えば GlassGiant.com[3] はユーザが Web 上で送信した画像をアスキーアートに変換する CGI プログラムであり，他にも TEXT-IMAGE.com[4] などが挙げられる．後者では文字の色を変化させることでカラーのアスキーアートを作成することも可能となっている．また，Give me chocolate[5] は同様の手法で作成されたアスキーアートをメールでユーザに送信する機能を備えている．これらの手法で生成されたアスキーアートは，視覚的に楽しむことは可能であるが，文字列自体は文章として意味を成さない．本論文では文字の間隔を変化させることで画像の濃淡を表現しており，入力に用いる文字列は元の順番を保ったままであることから，アスキーアートを文章として読むことも可能である．

図 3 Popular Science Magazine 1939 年 6 月号に掲載されたアスキーアート[1)2)]
Fig. 3 An ASCII-art in the Popular Science Magazine, June 1939[1)2)]

複数の記号やアルファベットから構成される「顔文字」や，Web 上の掲示板などで用いられることが多い，複数行を用いたキャラクター表現もアスキーアートと呼ばれることがある．これらは文字を構成する線に着目して，特定の線画を表現したものと言うことができ，そのようなアスキーアートを多数まとめた本も出版されている[8]．また，この手法を用いて作られたアスキーアートを多数用意し，一定時間間隔で表示を切り替えることでアニメーションを実現しているものもある[6)7)]．この手法で作られたアスキーアートは，本稿で対象としている濃淡画像を表現するものと比較して，少ない文字数で形を表現できる利点がある一方で，計算機による自動生成が難しく，綺麗で小さな形を作るためには人間が手動で文字を選択・修正しなければならないといった問題点がある．また，この手法で作成されたアスキーアートも，文字の並びにコンテキストを持たせることはできない

意味のある文字列の並びで視覚的な表現を行っている例としては，IBM の Visual Communication Lab による Many Eyes[9] がある．このシステムでは入力されたテキストデータを解析し，単語の出現頻度に応じて大きさを変化させて出力する「Word Cloud」や単語の接続を表現した「Phase Net」などを自動で生成し，結果を Web で共有することができる．このシステムは本研究とは異なり，文字が単語単位でしか接続していないため文章としては成立していないことと，情報の視覚化という目的に特化しているため，その結果をユー

© 2010 Information Processing Society of Japan

ザがコントロールすることは難しいという違いがある．文章と絵を重ね合わせ各画素に対して乗算処理を行うことで可読性を備えたアスキーアートを生成した例としては，Mozilla Foundation による NY Times の広告がある[10)11)]．この手法と比較し，我々の手法はよりノイズ成分が少ないなめらかな平面を表現することができ，また文字を詰めることでより濃い画像を表現することが可能である．

図 4 文字列と画像の輝度値を乗算した例[10)]
Fig. 4 An example of ASCII-art in which brightness of the image and the texts are multiplied[10)].

3. システムの詳細

アルファベットやひらがななど，画数の少ない文字の並びは視覚的に薄く認識され，画数の多い漢字などの文字の並びは濃く認識されるという特徴がある．本システムではその性質を濃淡画像を表現するアスキーアートに利用する．

ユーザーは，アスキーアートで表現する濃淡画像と文章の記述されたテキストファイル，および行の高さ（ピクセル数）を入力する．入力画像はグレースケール 256 階調のラスター画像で，もしそれ以外の形式の画像を入力した場合は自動的に条件に合う画像へと変換される．入力に対してシステムは各文字ごとの字間を計算し，その結果を SVG または PNG 形式のアスキーアート画像として出力する．以下にシステムの流れを示す．

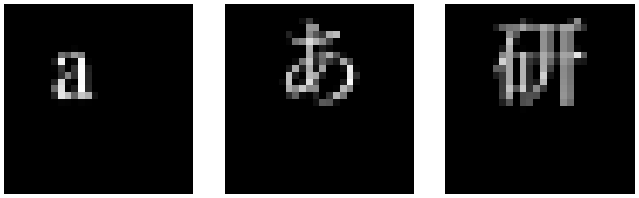
(1) 画像ファイル，テキストファイル，行の高さを読み込む
(2) テキストファイルに現れる各文字の濃度を計算する
(3) 入力画像の明るさ，コントラストを調整する
(4) 画像のサイズ調整を行う
(5) 各文字列の位置を計算する
(6) 求められた位置に合わせて文字列を出力する

この流れで行われる処理の詳細を以降で述べる．

3.1 積算画素濃度の計算方法

文字はその画数・形状，およびサイズによって，描画した際に文字を構成する画素数が異なる．本論文ではアンチエイリアスを有効にした状態で，ある 1 文字を描画した際の，各画素の濃度（0/255〜255/255）の合計値を，その文字の「**積算画素濃度**」と呼び，入力文字列の先頭から i 文字目の文字の積算画素濃度を $B_{\mathrm{char}}(i)$ と表記することとする．各画素の持つ濃度は完全な黒の場合は 1，白の場合は 0 であり，$B_{\mathrm{char}}(i)$ はそれらを合計したものであるため 0 以上の実数値である．また積算画素濃度の値を文字のバウンディングボックスの面積で割ったものを「**文字濃度**」と呼ぶ（文字濃度は 0.0〜1.0 の値を取る）．

最初に，文字がはみ出さない程度の大きさの 8bit グレースケールのキャンバスを用意する．本システムでは，一辺の長さを与えられたフォントサイズ の 2 倍とした．指定されたフォントで，アンチエイリアスを有効にした状態で，キャンバスに一文字をレンダリングする．各画素の濃度を左上から順に走査し，すべての画素の濃度の和を，その文字の積算画素濃度とする．この処理を与えられたテキストファイルの最初の文字から順に実行していく（**図 5**）．



16.11　　31.03　　48.07

図 5 文字のレンダリング結果と積算画素濃度の値．図は Times 書体を用い，14px で文字をレンダリングし積算画素濃度を取得した例．キャンバスサイズは 28px 四方．
Fig. 5 Images on which a character is rendered. The numerical values are summation of the darkness of each pixel. 14 pixel sized “Times” font is used. The size of canvas is 28 pixel square.

なお，本プログラムでは高速化のために，文字のレンダリングは黒の背景に白色で行うこ

© 2010 Information Processing Society of Japan

とで，輝度値を濃度として算出することを行った．また，同じく高速化のために文字コードをキー，積算画素濃度を値としたハッシュテーブルに値をキャッシングし，同一文字の積算画素濃度を複数回参照する際はキャッシュのデータを用いることとした．

3.2 入力画像のサイズ調整

多くの場合，画像の濃度に合わせて単純に文字を配置していくと，画像から文章があふれてしまったり，画像の途中で文章が終わってしまったりする．そこで本システムでは画像の大きさを調整し，入力テキストの濃度値と画像の濃度の総量が一致するようにする．そのときの入力画像の拡大率 s は次のように求める．

$$s = \sqrt{\sum_{\text{all}} B_{\text{char}} \Big/ \sum_{\text{all}} D_{\text{image}}} \tag{1}$$

ここで，D_{image} は入力画像のピクセル濃度を表す．しかしながら，この値を直接用いて画像を拡大すると画像の高さが H_{line} の整数倍ではなくなってしまう場合がある．この場合も画像が大きすぎて文章が足りないという状況と同様の結果となってしまうため，画像の大きさを変更した後，入力画像の縦横比を調整し，面積をほぼ保ったまま画像の高さが H_{line} の整数倍となるようにする．入力画像の幅，高さをそれぞれ W_{in}，H_{in}，行の高さを H_{line} で表現し，変形後の画像のサイズ W'_{in}，H'_{in} は次の式で求める．

$$H'_{\text{in}} = \lceil sH_{\text{in}} - (sH_{\text{in}} \bmod H_{\text{line}}) \rceil \tag{2}$$

$$W'_{\text{in}} = \left\lceil \frac{s^2 W_{\text{in}} H_{\text{in}}}{H'_{\text{in}}} \right\rceil \tag{3}$$

式 2 では s をかけて変形させた後の画像の高さから余分に伸びてしまった分 ($sH_{\text{in}} \bmod H_{\text{line}}$) を取り除いている．また，式 3 では H'_{in} にあわせて，面積が変化しないよう値を計算している．この処理を適用すると，画像のアスペクト比が変化する．拡大後の画像の高さ H'_{in} が 500px で 12px の場合，縦方向の余剰な伸びの大きさは $500 \bmod 12 = 8$px となるが，その割合は元の高さの値の 1.6%でしかない．文章が十分に長く画像が十分拡大されればアスペクト比の大きさは相対的に小さなものとなるため，無視できるものと見なすことができる．また，それぞれの式において値の切り上げを行っているため面積はわずかに増加し最後の行の末尾は数文字足りないものとなってしまうが，それによる画素の空きは文末位置の調整によって処理する．切り上げを行わないと，場合によっては文字が余ってしまい，末尾に数文字だけで構成された行が発生してしまう．それと比較し，数文字足りない行を自然に処理することは容易である．

```
画像の積算濃度値 = 0
文章の積算濃度値 = 0
i = 0
foreach(l = 0.. 行数) {
 foreach(x = 0.. 入力画像の幅) {
  画像の積算濃度値 += l 行目の左から x ピクセル目に相当する画素 H_line px 分の濃度値の和
  while(文章の積算濃度値 + B_char(i) > 画像の積算濃度値) {
   p(i) = (x, l * H_line )
   文章の濃度積算値 += B_char(i)
   i++
  }
 }
 行末のはみ出しを修正
}
```

図 6 文字の位置を算出するための疑似コード
Fig. 6 Pseudo code for calculating the positions of characters.

3.3 文字の位置の決定

与えられた文字列について，それを構成する各文字の位置は，画像の左上から一行ずつ順に決められる．まず最初に，システムは画像を複数の行に分ける．行の高さは H_{line} ピクセルとすると，行数は $H'_{\text{in}}/H_{\text{line}}$ となる．目的は文字列を構成する文字 C(i) (i=0...n-1) がどこに配置されるかを算出することであるが，文章の先頭から第 i 番目の文字 C(i) の左上隅の座標 p(i) は，**図 6** の擬似コードによって求められる．

なお，空白文字やカンマ，ドットなど積算画素濃度が小さな文字の場合は同一の場所に完全に重なった状態で描画される場合がある．また，そのままでは文字がもともと持っている幅を考慮していないため，各行の文章末尾の位置が最大で末尾の文字の字幅分だけはみ出してしまう．それを整え，行の末尾と文章幅が一致するようにするために，各行においてはみ出した量を均等に各文字の間隔に割り振って調整する処理を行う．

4. 入力画像の明るさ・コントラストの調整と可読性

本章では人間の目で文字の並びを見た際にどれだけ内容を理解することが可能かどうか

© 2010 Information Processing Society of Japan

Vol.2010-CG-139 No.3
2010/7/16

という，可読性について考える．一般に文字の重なりが大きければ大きいほど文字列の内容を認識することは難しくなり，可読性は失われる．**図 7** は実際に文字を重ねた様子であるが，この図から人の目による可読性を維持するには，重なりの量を 50%前後としておくことが望ましいといえる．

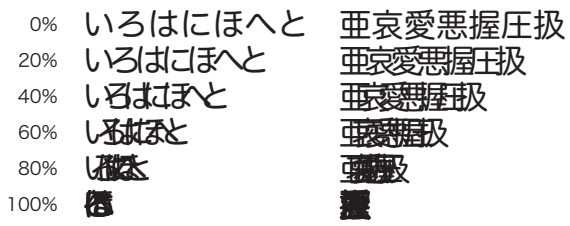


図 7 文字の重なりの量による見た目の違い
Fig. 7 The appearances of characters with different amount of overlaps.

また，**表 1** は，提案手法を用いて計測した文字濃度の平均値である．書体のサイズは 14px とした．この結果から，平均的な文字濃度は約 0.2 程度であることが分かる．

表 1 各文字の濃度．書体のサイズは 14px である
Table 1 The darkness of the charactor. The font size is 14px

	MS PGothic	MS PMincho
アルファベット（小文字）	0.20	0.13
アルファベット（大文字）	0.22	0.14
ひらがな	0.20	0.11
カタカナ	0.18	0.09
CJK 統合漢字（20991 文字）	0.32	0.20

上記の結果より,「人の目」による可読性をそなえたアスキーアートを目的とする場合は，入力画像に 0.2 / 0.50 = 0.4 以上の濃度の画素を用いないようにすべきであると言える．**図 8** は実際の可読性を評価するために指定された濃度となるよう「いろはうた」を 10 回（470 文字）ならべたものであるが，この実験から，画素の密度が 0.6 以上の場合は人間の目で文字列の並びを認識することは難しいと言える．本プログラムでは入力画像に対して濃度の最大値・最小値に制限を加え全ての画素の濃度が指定の範囲内になるように輝度の値を正規化する機能が実装されており，ユーザは任意でその機能を使用することが可能である．

なお，図 8 では書体に MS Mincho 14px を，H_{line} には 10px を指定している．

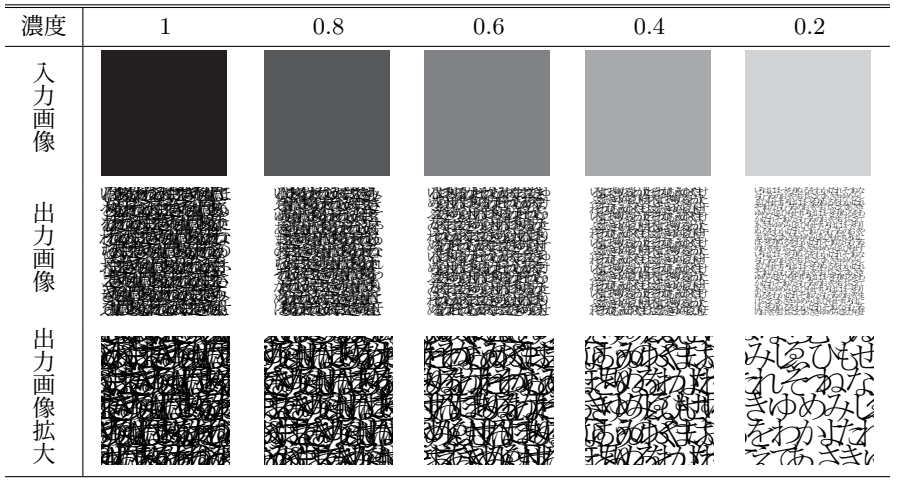


図 8 文字の重なりを変化させながら「いろはうた」を並べた例
Fig. 8 The appearances of characters arranged so that the density matches to specified value.

しかしながら本システムの場合は SVG 形式の画像で結果を出力することもが可能となっており，その場合は使用されている文字列を容易に取り出すことが可能なことから，作品を電子的なデータの配布に限定する場合はそのような制約は無視できるだろう．実際に本論文 5 章に掲載されているアスキーアート画像は文字情報を保持しており，図中のテキストを選択してコピーしエディタなどにペーストすると，元となる文章が現れる．

なお，入力画像に対して出力画像の濃度が全体的に低くなっているのは，文字列の重なりの影響によるものであると考えられる．本手法では文字というすでに決まった形のものを重ねるため隙間ができてしまい，どうしても元の画像よりもコントラストが小さなものとなってしまう．そのためユーザはそれを見越して入力画像を編集し，コントラストを高いものへと修正しておく必要があるだろう．

5. 作　　例

提案手法を用いて作成された作品の作例および文章の量，計算に要した時間を**図 9** ～ **図 12** に示す．実行環境には MacBook Pro (15inch, Late 2009) [CPU: Intel Core2 Duo

© 2010 Information Processing Society of Japan

Vol.2010-CG-139 No.3
2010/7/16

2.66GHz, RAM: 4GB, MacOS 10.6.5 with Java 1.6.0_20] を利用した．提案手法を用いたことで，モノの画像や政治家・作家の肖像を、それと関連の深い文章で表現することが可能となり，画像により深い意味を直接持たせることが可能となった．なお，これらの画像上をドラッグしコピーを行うことで，アスキーアートを構成している文字を取り出すことが可能となっている．

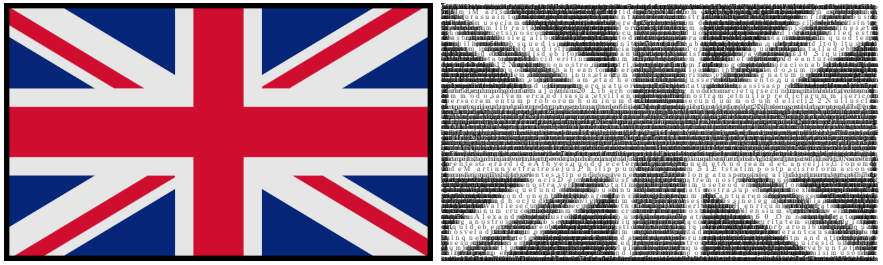
6. まとめと課題

本研究では可読性を備えたアスキーアートの生成手法を提案し，実装を行った．その際に必要となる，文字の濃度の計算手法と文字の位置の計算方法，入力として与えた文字列を過不足なく使用して画像を表現するための倍率の計算手法について述べた．また，出力されるアスキーアートを，人間の目でも読めるようにする際の注意点についても述べた．そして複数の例題について実際に生成してみることで，本手法でアスキーアートの制作が可能なことが確認できた．

なお，本手法で生成される画像の濃度は入力画像に対して全体的に低くなってしまうという問題点がある．そのため，入力画像に対してシステムが自動的にコントラストを調整し，元画像の濃度に近づくように修正を加える機能があると良いだろう．また，綺麗なアスキーアートを生成するためには十分な解像度が必要で，文字数が少ないと出力結果がぼやけてしまって認識しづらくなってしまう．これは画像処理に関する一般的な問題であるが，本研究の場合は文字の角度を調整し，文字のパーツと元画像の輪郭を一致させることで，見た目の解像感が高まる可能性があると考えられる．より少ない手順で美しい出力結果を得られるようにすることが今後の課題である．

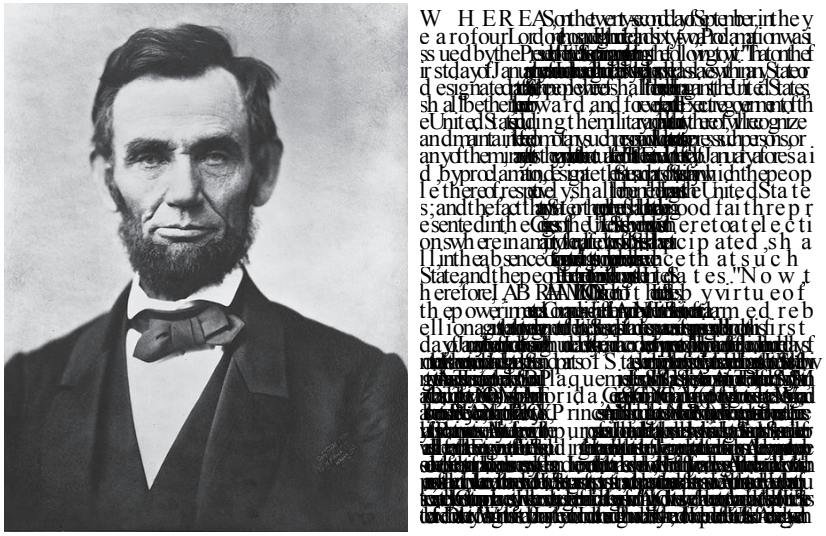
参 考 文 献

1) Rosaire J. Belanger, “Typewriter Artist Produces Pictures Like Tapestry”, Popular Science Magazine, June 1939
2) Cumbrowski, Carsten (2007-02-14), Keyboard Text Art From Over Twenty Years Before ASCII, roysac.com, retrieved 2008-03-05, http://www.roysac.com/blog/2007/02/keyboard-text-art-from-over-twenty-years-before-ascii/
3) Shawn Wilson, GlassGiant.com, http://www.glassgiant.com/ascii/
4) Patrik Roos, TEXT-IMAGE.com, http://www.text-image.com/index.html
5) MAKIBISHI INC. , Give me chocolate -Image to HTML e-mail, http://chocolate.makibishi.co.jp/
6) STAR ASCIIMATION WARS, telnet towel.blinkenlights.nl



Text length: 24,998, Time to calcurate text positons: 57ms

図 9 イギリス憲法の大憲章で描画された英国国旗
Fig. 9 The image of the Union Jack consisted with the Magna Carta.

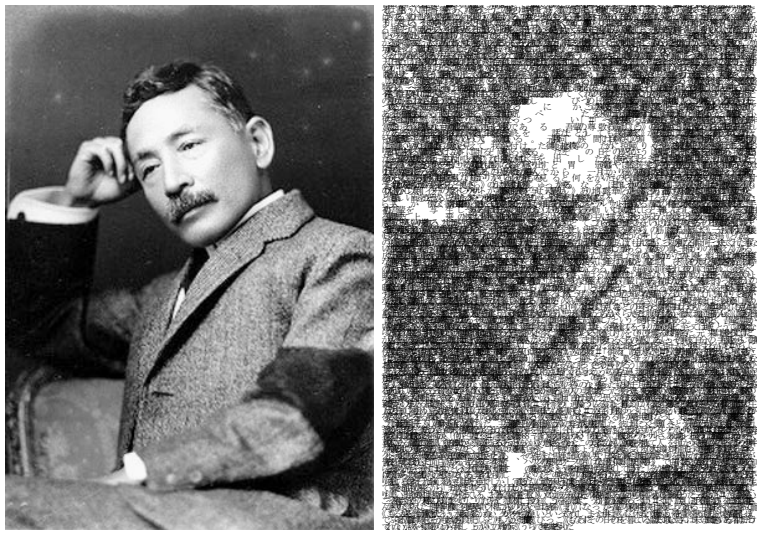


Text length: 4,229, Time to calcurate text positons : 14 ms

図 10 ゲティスバーグ演説で描画されたリンカーンの肖像
Fig. 10 The portrait of Lincoln consisted with The Gettysburg Address.

© 2010 Information Processing Society of Japan

Vol.2010-CG-139 No.3
2010/7/16

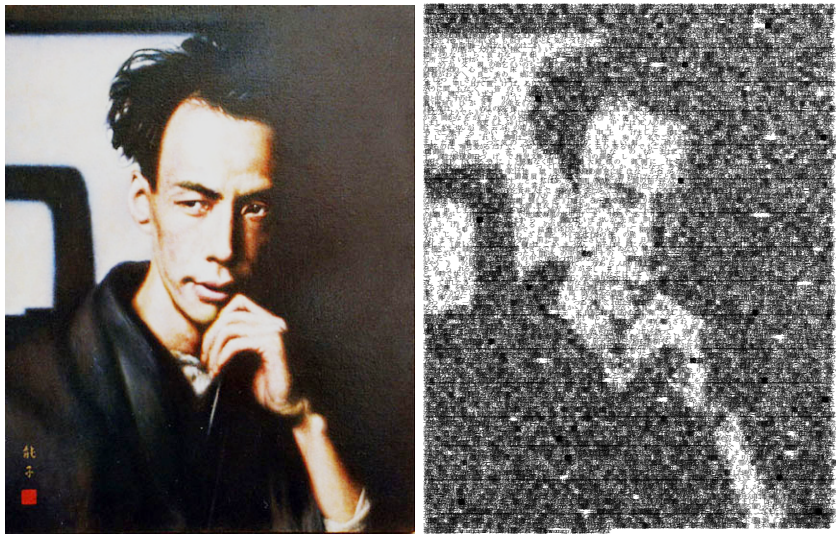


Text length: 13,979, Time to calcurate text positons: 65 ms

図 11 「吾輩は猫である　第一章」で描画された夏目漱石

Fig. 11 The portrait of Soseki NATSUME consisted with the 1st chapter of his novel, “I Am a Cat”.

7) Katamuki, 新世紀エヴァンゲリオン OP::Ascii Art Anime, http://asciiartanime.com/gallery/index12.html
8) まるへそ太郎 , 2 ちゃんねる:「2 ちゃんねる AA 大辞典」, ソフトバンククリエイティブ, ISBN: 4797322756, 2003/2/28
9) IBM Visual Communication Lab, “Many Eyes beta for shared visualization and discovery”, http://manyeyes.alphaworks.ibm.com/manyeyes/
10) designed by Christopher Messina, Mozilla Foundation: “Firefox 1.0”, New York Times, pp24-25, 2004 Dec 16, http://mozilla.jp/press/releases/2004/12/15/
11) Yoichi YAMASHITA: “FireFox, NY Times 紙でサポーターの名前を使った全面広告”, マイコミジャーナル, 2004 Dec 17, http://journal.mycom.co.jp/news/2004/12/17/100.html



Text length: 29,841, Time to calcurate text positons: 48 ms

図 12 「地獄変」で描かれた芥川竜之介

Fig. 12 The portrait of Ryunosuke AKUTAGAWA consisted with his novel, “Hell Screen”.

© 2010 Information Processing Society of Japan