

Character Recognition by Feature Point Extraction

Eric W. Brown
feneric@ccs.neu.edu
Copyright © 1992

Abstract

The ability to identify machine printed characters in an automated or a semi-automated manner has obvious applications in numerous fields. Since creating an algorithm with a one hundred percent correct recognition rate is quite probably impossible in our world of noise and different font styles, it is important to design character recognition algorithms with these failures in mind so that when mistakes are inevitably made, they will at least be understandable and predictable to the person working with the program. This paper explores one such algorithm and tests it on two different fonts using a third font as a reference. The results are discussed and several improvements are suggested.

Keywords

Text recognition, Optical character recognition, Feature extraction, Pattern recognition, Feature point, OCR, Off-line character recognition, machine printed character recognition.

Optical character recognition (OCR) has been a topic of interest since possibly the late 1940's when Jacob Rabinow started his work in the field¹. The earliest OCR machines were primitive mechanical devices with fairly high failure rates. As the amount of new written material increased, so did the need to process it all in a fast and reliable manner, and these machines were clearly not up to the task. They quickly gave way to computer-based OCR devices that could outperform them both in terms of speed and reliability.

Today there are many OCR devices in use based on a plethora of different algorithms. All of the popular algorithms sport high accuracy and most high speed, but still many suffer from a fairly simple flaw: when they do make mistakes (and they all do), the mistakes are often very unnatural to the human point of view. That is, mistaking a "5" for an "S" is not too surprising because most people are willing to agree that these two characters are similar, but mistaking a "5" for an "M" is counter-intuitive and unexpected. Algorithms make such mistakes because they generally operate on a different set of features than humans for computational reasons. Humans observe strokes and the relations between them, while algorithms measure anything from Transformation Ring Projections² of a character to the Fourier Transform of the Horizontal-Vertical Projections³ of a character. These methods do work and are often computationally efficient, but they make the computer see letters through a decidedly non-human set of eyes.

The importance of making the same sorts of mistakes as a human may not be immediately obvious, but it is important to realize that the main purpose of OCR is to facilitate communication between humans. Mistakes typical of humans can be more readily corrected by humans (i.e. "5ave" is easier to connect with "Save" than "Mave").

This paper describes an algorithm that attempts to work with a subset of the features in a character that a human would typically see for the identification of machine-printed English characters. Its recognition rate is currently not as high as the recognition rates of the older, more developed character recognition algorithms, but it is expected that if it were expanded to work with a larger set of features this problem would be removed. If it were expanded to use more features, it would be made correspondingly slower; with the advent of faster microprocessors this fact is not viewed as a crippling problem.

The characters in most Western character sets can comfortably be described using only an eight by eight grid. Since the focus of this algorithm is currently just machine printed English characters the algorithm was designed to handle eight by eight data. It should not be impossible to use this algorithm on characters with higher resolution, but some method of data reduction would have to be applied to the raw data first. A block reduction method should produce acceptable results for most machine printed data, but there is no reason that a good thinning algorithm⁴ could not do just as well provided that the input needs of this algorithm are properly understood.

The algorithm presently avoids thinning (and other preprocessing) by assuming that the input eight by eight data is not particularly aberrant -- lines in an eight by eight grid should not normally be thicker than two pixels. With this assumption, it then proceeds to look for feature points.

A feature point is a point of human interest in an image, a place where something happens. It could be an intersection between two lines, or it could be a corner, or it could be just a dot surrounded by space. Such points serve to help define the relationship between different strokes. Two strokes could

fully cross each other, come together in a "Y" or a "T" intersection, form a corner, or avoid each other altogether. People tend to be sensitive to these relationships; the fact that the lines in a "Z" connect in a certain way is more important than the individual lengths of those lines. These relationships are what should be used for character identification, and the feature points can be exploited for the task.

The procedure for extracting these feature points utilized by this algorithm is fairly straightforward. Since an eight by eight character consists of only sixty-four pixels, it is viable to simply loop through the entire character and examine each pixel in turn. If a pixel is on, its eight neighbors are checked. Since each neighbor can also only be on or off, there are merely 256 possible combinations of neighborhoods. Of these 256, fifty-eight were found to represent significant feature points in a fairly unambiguous way. Extracting feature points thus reduced to calculating a number between zero and 256 to describe a pixel's neighborhood and then comparing that number against a table of known feature points (see Table 1, "Enumeration of Possible Pixel Neighborhoods,"). While it is true that this method does not always catch every feature point (some can only be seen in a larger context) it catches the majority. Missing feature points is certainly not a limiting factor in the algorithm's accuracy. It also does not suffer from labelling too many uninteresting points as being feature points. It has virtually no false positives. The feature point extractor is thus fast and reliable.

Table 1: Enumeration of Possible Pixel Neighborhoods

+	+ *	+ *	+ **	*+	*+ *	*+ *	*+ **	*+	*+ *	*+ *	*+ *	*+ *	**+	**+	**+
0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
*+	*+ *	*+ *	*+ **	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+
1	17	33	49	65	81	97	113	129	145	161	177	193	209	225	241
*+	*+ *	*+ *	*+ **	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+
2	18	34	50	66	82	98	114	130	146	162	178	194	210	226	242
**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+	**+
3	19	35	51	67	83	99	115	131	147	163	179	195	211	227	243
+*	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**
4	20	36	52	68	84	100	116	132	148	164	180	196	212	228	244
+	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*	*+*
5	21	37	53	69	85	101	117	133	149	165	181	197	213	229	245

Table 1: Enumeration of Possible Pixel Neighborhoods

+	*+* *	*+* *	*+* **	*+*	**+*	**+*	**+*	**	**	**	**	**	**	**	**
6	22	38	54	70	86	102	118	134	150	166	182	198	214	230	246
**+*	**+* *	**+* *	**+* **	**	**	**	**	**	**	**	**	**	**	**	**
7	23	39	55	71	87	103	119	135	151	167	183	199	215	231	247
+*	+**	+**	+**	+*	+**	+*	+*	+*	+*	+*	+*	+*	+*	+*	+*
8	24	40	56	72	88	104	120	136	152	168	184	200	216	232	248
+	*+ **	*+* *	*+ ***	**+ *	**+ **	**+ **	**+ ***	**+ *	**+ **	**+ **	**+ ***	**+ *	**+ **	**+ **	**+ ***
9	25	41	57	73	89	105	121	137	153	169	185	201	217	233	249
+	*+ **	*+* *	*+ ***	**+ *	**+ **	**+ **	**+ ***	**+ *	**+ **	**+ **	**+ ***	**+ *	**+ **	**+ **	**+ ***
10	26	42	58	74	90	106	122	138	154	170	186	202	218	234	250
+ *	**+ **	**+ **	**+ *	**+ *	**+ **	**+ **	**+ ***	**+ *	**+ **	**+ **	**+ ***	**+ **	**+ **	**+ **	**+ ***
11	27	43	59	75	91	107	123	139	155	171	187	203	219	235	251
+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**	+**
12	28	44	60	76	92	108	124	140	156	172	188	204	220	236	252
+ *	*+* **	*+* **	*+* ***	*+* *	*+* **	*+* **	*+* ***	*+* *	*+* **	*+* **	*+* ***	** *+**	** *+**	** *+**	** *+**
13	29	45	61	77	93	109	125	141	157	173	189	205	221	237	253
+ *	*+* **	*+* **	*+* ***	*+* *	*+* **	*+* **	*+* ***	** +**	** +**	** +**	** +**	** *+**	** *+**	** *+**	** *+**
14	30	46	62	78	94	110	126	142	158	174	190	206	222	238	254
+* *	**+* **	**+* **	**+* *	** *+**	** *+**	** *+**	** *+**	** +**	** +**	** +**	** +**	** *+**	** *+**	** *+**	** *+**
15	31	47	63	79	95	111	127	143	159	175	191	207	223	239	255

Characters cannot be identified by the extraction of feature points alone. Without a database of characters and their associated feature points, the ultimate feature point extractor would be useless. Only with such a database can the feature point extraction results from an unknown character be compared

against what is expected for real-world characters and a judgement of the unknown's identity made. Thus a "gold-standard" dictionary of characters and their associated features must be defined. Ideally this dictionary should contain details for the average appearance of every character manifestation (many English characters have multiple different accepted manifestations -- note "Z" versus "Z̄"). If poor representatives appearances for characters are chosen valid characters at the extremes will not be identified as readily. If some manifestations of characters are missed, the program will certainly not be able to identify characters belonging to these groups at all.

With both a method for extracting feature points and a dictionary of characters and associated feature point data for reference, identifying characters becomes a problem of measuring the degree of similarity between two sets of features. The method employed by this algorithm is just a slight modification of Euclidean distance. All the distances between each of the feature points in the unknown character and their closest corresponding feature points in the reference character are summed and missing or extra feature points are penalized. Identification is then a matter of finding the character in the dictionary that is within a certain threshold distance of the unknown character. In practice, the algorithm currently checks every character in the reference set to first locate the minimum distance, and then verifies that the minimum distance is less than the threshold. Additionally, the algorithm tries to make some simple compensation for noise by noting that pixels surrounded by completely empty space (*dots*) and pixels surrounded by completely full space (*blots*) are quite uncommon in normal characters and are probably the result of some type of noise in the input.

Figure 1: Character Display

```
(D)isplay char, (R)un new set, or (Q)uit: d
Character number: 68
Character name: K

  XX *X
  ** **
  ****
  **X
  ****
  ** **
  XX *X
```

The algorithm was run using three different sets of input data. The first was obtained from the ROM of an old CBM C128 computer⁵ (see Figure 2, "The CBM Character Set & Extracted Feature Points,"). This set was used to build the dictionary in spite of the fact that it really does not adequately provide all the information required for a good gold standard. It has just one manifestation for each character, and the appearance portrayed for each is often somewhat different than what would be expected for an average representative of the type. The other two character sets (see Figure 3, "The 'Ult' Character Set & Extracted Feature Points," and Figure 4, "The 'Alt' Character Set & Extracted Feature Points,") were pulled from old C128 software and were used for comparison against the ROM set. They served as a sample set of unknowns. Additionally the CBM set was run against itself. Each character set contained eighty-four different characters consisting of the uppercase letters, the lowercase letters, the numbers, a few mathematical symbols, and several punctuation marks. The output for each set run against the reference consisted of ASCII output showing the shape of the charac-

ter (with the feature points marked), the name of the closest match, the distance from that closest match, and the name of the character if identification was possible. Characters could also be displayed from the dictionary from a menu prompt to make visual comparison of the assorted characters in the test sets and the dictionary characters possible (see Figure 1, "Character Display,").

Running the dictionary set against itself produced no useful information (as may have been expected). In every case the unknown character matched itself without difficulty. Running against the other sets proved to be more interesting. The results are detailed in Table 2, "Character Recognition Results," and in the Appendix but in general they did make numerous mistakes. More than half the mistakes were understandable considering the difference in font styles (the "7" in the 'Ult' font really does look a lot like the "?" in the CBM font and the "J" in the 'Ult' font is not too far off from a "U") or general character appearance (distinguishing between a "l" and an "I" or a "0" and a "O" sometimes has to be done by context even for humans) or due to different manifestations of the same character (the "a", "z", "Z", and "4" in the 'Ult' font were less common varieties). The remainder were understandable when the nature of the features was considered. These mistakes provide the information needed to improve the algorithm.

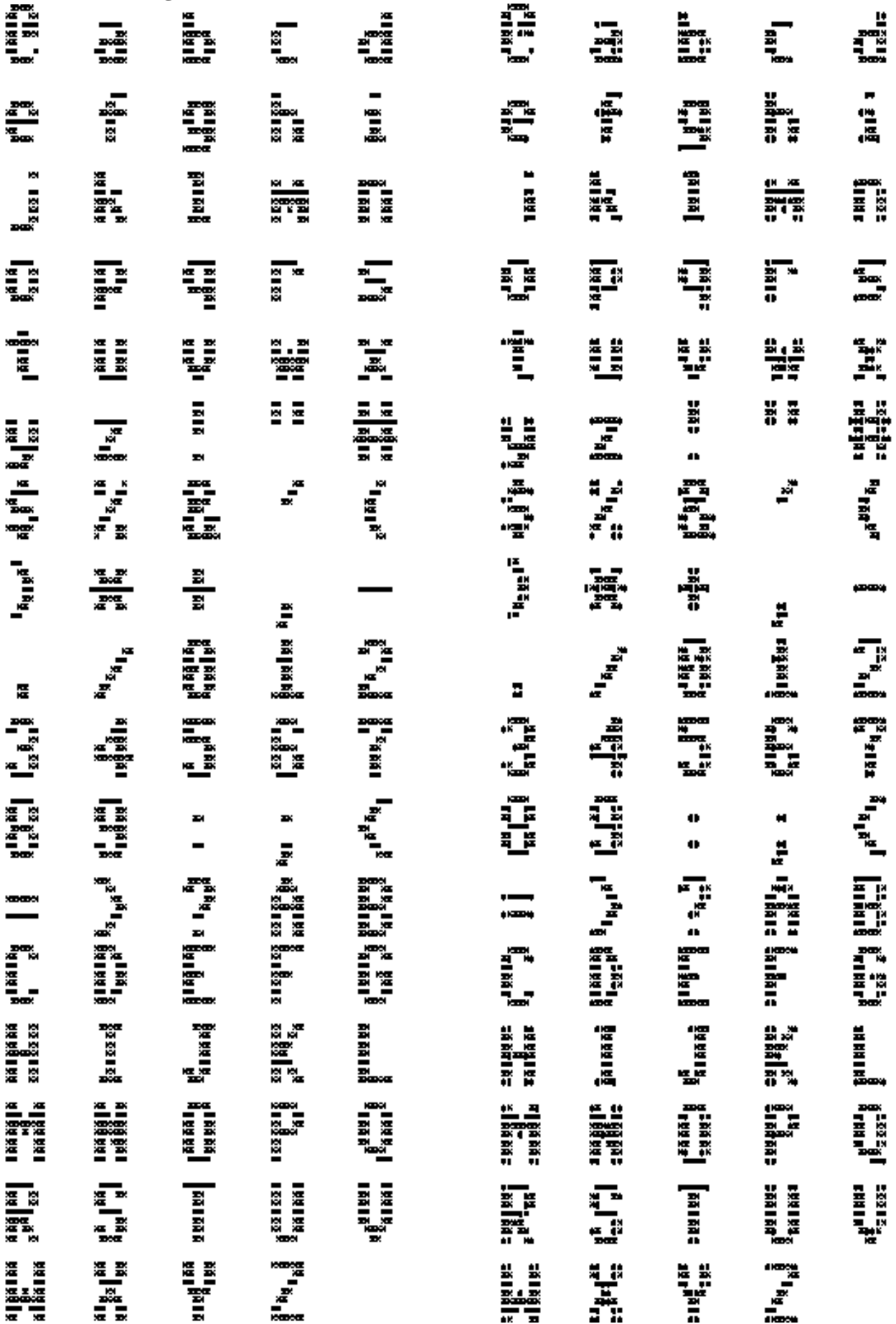
Table 2: Character Recognition Results

Table 2:

Results for the 'Alt' Character Set		
Total Correct	72	86%
Total Correct (without counting identical characters)	21	25%
Total Unknowns	5	6%
Total Wrong Guesses	7	8%
Results for the 'Ult' Character Set		
Total Correct	49	58%
Total Correct (without counting identical characters)	28	33%
Total Unknowns	12	14%
Total Wrong Guesses	23	27%
Total Combined Results		
Total Correct	121	72%
Total Correct (without counting identical characters)	49	29%
Total Unknowns	17	10%
Total Wrong Guesses	30	18%

Several improvements are possible. In the current algorithm, the different types of feature points ("T" intersections, "Y" intersections, corners, crosses, ends, etc.) are all treated as generic feature points. This approach was taken because not every type of feature point can be unambiguously defined for a given eight pixel neighborhood about a central pixel. Considerable logic would be needed to resolve the bulk of these ambiguities based upon the other surrounding feature points. Nonetheless, resolution for most of them is possible and would provide a great deal more information about each character. If this were done, a "c" would no longer be mistaken for an "o" for example.

Figure 2: The CBM Character Set & Extracted Feature Points



It would also be possible to examine the space between individual feature points to determine whether or not they were connected by contiguous straight lines. This would also greatly enhance the accuracy of the algorithm and would prevent a "W" from being recognized as an "E". This particular modification would both slow the algorithm down considerably and consume quite a bit more memory, but it could still be justifiable if the accuracy increase were significant.

A line and/or curve extractor could be used in conjunction with (or independently of) the above mentioned modification, and would provide yet more usable features that could be exploited for identification.

Further, many refinements of the current thresholds and modifiers could be made to the algorithm to improve performance. It could easily be better tuned to the data used here, but better results through such tweaking really was not the intent of this work.

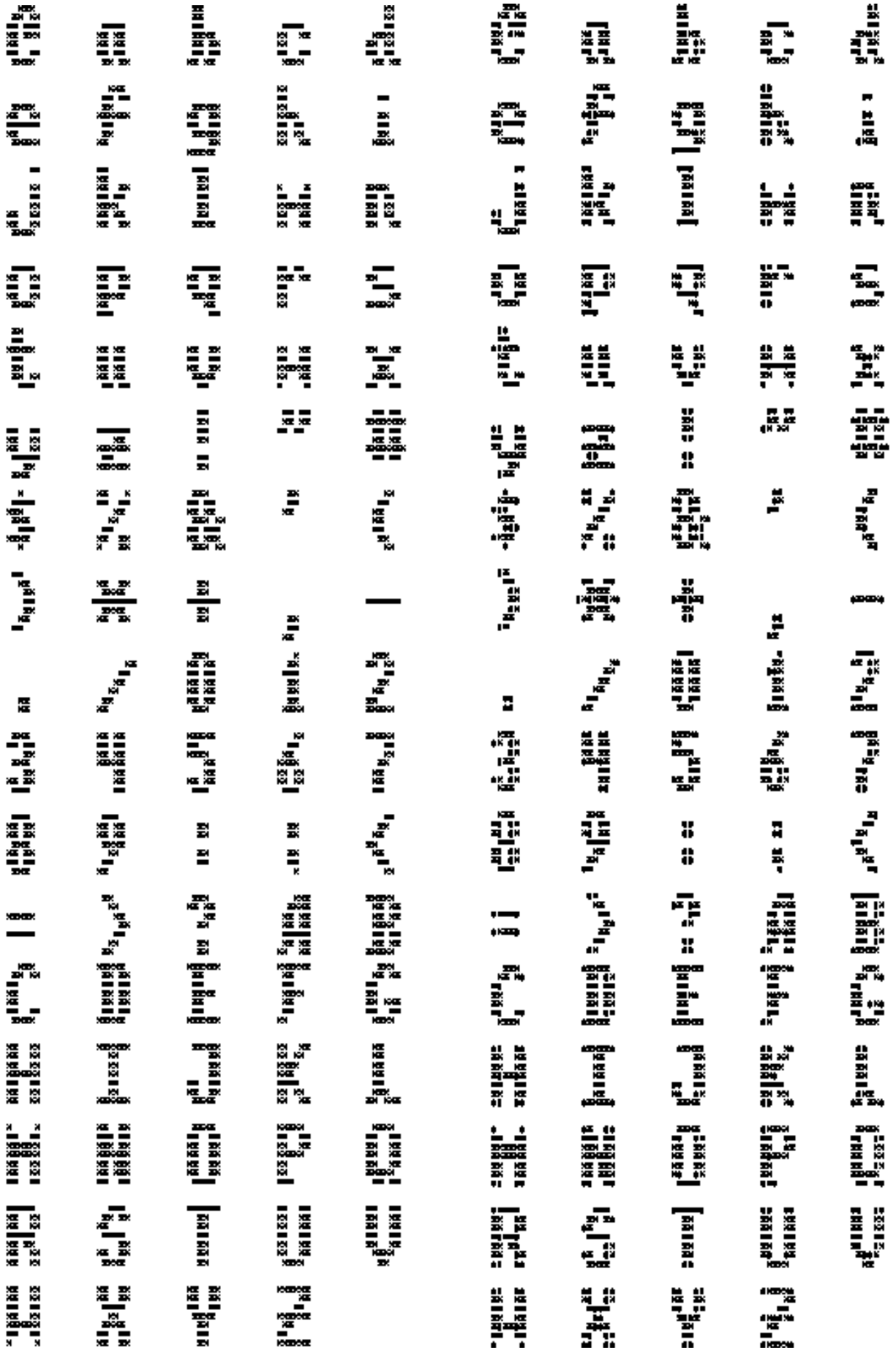
The exploration of modifications that would make the algorithm more invariant to translation would certainly be useful. If the use of lines connecting feature points as described above provided enough information in and of itself to accurately identify characters, this would be preferable to the current method of using feature point location as it would be translation invariant while the current method is clearly not. Even if dependence on location cannot be fully removed, it could be reduced through a separate preprocessing step. Each character could be centered in the eight by eight grid with special attention being paid so position information is not lost on characters where such information is vital (the comma and apostrophe, for example).

Numerous other little miscellaneous improvements could be made to various features of the algorithm. The noise detection / handling procedure is currently little more than a stub and could be readily improved. The character dictionary could be sorted in order of frequency and the thresholds trusted more completely to improve overall speed (the current algorithm is fairly quick anyway; on a Sun SPARCstation SLC the initial building of the dictionary took less than a second and the running of other character sets was held up by screen output). Changes like these are mostly cosmetic however and have little bearing on the true algorithm.

Suggested future work includes both the testing of these algorithm changes and further testing of the algorithm with more character data. Of particular interest would be character data that is deliberately noisy and character data that has been reduced to eight by eight resolution from some greater resolution. Both of these cases reflect real-world problems.

Overall the results of this experiment were mixed. On the one hand, the initial results certainly are not of commercial quality. When only a couple of pixels differed between the unknown character and the reference, the results were fairly good, but larger differences often made the algorithm unable to correctly identify the unknown character. On the other hand, the low success rate is not indicative of the general algorithm, just the current implementation. There are many possible changes that could vastly improve the algorithm's recognition abilities. With a few of these changes implemented, the mistakes the algorithm would make would indeed be very similar to the types of mistakes humans would make. Thus general algorithm holds promise as a character recognizer that identifies characters in a manner similar to the way that humans identify characters.

Figure 3: The 'Ult' Character Set & Extracted Feature Points



References

1. Personal conversation with Dana Sawyer, who indirectly knew Jacob Rabinow during the 1960's.
2. Y. Y. Tang, H. D. Cheng, & C. Y. Suen, "*Transformation-Ring-Projection (TRP) Algorithm and Its VLSI Implementation*", *Character & Handwriting Recognition: Expanding Frontiers*, 1991, World Scientific Publishing Co. Pte. Ltd., Singapore
3. N. G. Bourbakis & A. T. Gumahad II, "Knowledge-Based Recognition of Typed Text Characters", *Character & Handwriting Recognition: Expanding Frontiers*, 1991, World Scientific Publishing Co. Pte. Ltd., Singapore
4. Louisa Lam, Seong-Whan Lee, & Ching Y. Suen, "Thinning Methodologies -- A Comprehensive Survey"; *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 9, September 1992
5. The Commodore 128 Computer, Commodore Business Machines, 1984

Appendix

Full results for 'Alt'.chr

Cognition phase...

Done learning characters.

(D)isplay char, (R)un new set, or (Q)uit: r

Filename of set to be run: alt.char

Minimum distance: 0; Closest character: @

The character is a: @

Character name: @

```
****
*X **
** **
** X*X
**
*X *X
****
```

Minimum distance: 9; Closest character: d

The character is a: d

Character name: d

```
XX
**
**
***X*
*X **
*X *X
*** *X
```

Minimum distance: 4; Closest character: a

The character is a: a

Character name: a

```
X***
**
***X*
X* *X
*** *X
```

Minimum distance: 0; Closest character: e

The character is a: e

Character name: e

```
****
** **
*X***X
**
***X
```

Minimum distance: 14; Closest character: K

Unknown character.

Character name: ??

```
XX
**
**
*X***
** X*
X* X*
X* ***
```

Minimum distance: 13; Closest character: f

Unknown character.

Character name: ??

```
***
*X *X
**
XXX**X
**
**
XX
```

Minimum distance: 4; Closest character: o

The character is a: o

Character name: o

```
****
*X *X
**
*X *X
****
```

Minimum distance: 2; Closest character: g

The character is a: g

Character name: g

```
****X
*X **
*X **
***X*
**
X***
```

Minimum distance: 4; Closest character: h
The character is a: h
Character name: h

```
XX
**
**
*X***
** X*
** **
XX XX
```

Minimum distance: 2; Closest character: I
The character is a: I
Character name: I

```
X*X
**
**
**
**
**
X**X
```

Minimum distance: 0; Closest character: i
The character is a: i
Character name: i

```
XX

X*X
**
**
X**X
```

Minimum distance: 0; Closest character: m
The character is a: m
Character name: m

```
X* **
*****
**XXX**
** X **
XX XX
```

Minimum distance: 4; Closest character: j
The character is a: j
Character name: j

```
XX

XX
**
**
X* X*
****
```

Minimum distance: 10; Closest character: k
The character is a: k
Character name: k

```
X* ***
X* X*
** **
** **
XX XX
```

Minimum distance: 8; Closest character: k
The character is a: k
Character name: k

```
XX
**
** *X
** **
*X**
** **
XX *X
```

Minimum distance: 0; Closest character: o
The character is a: o
Character name: o

```
****
*X X*
** **
*X X*
****
```

Minimum distance: 13; Closest character: p
Unknown character.
Character name: ??

```
X* ***  
X* X*  
** X*  
*X***  
**  
XX
```

Minimum distance: 4; Closest character: q
The character is a: q
Character name: q

```
****X  
*X **  
*X **  
***X*  
*X  
*X
```

Minimum distance: 9; Closest character: r
The character is a: r
Character name: r

```
X* ***  
X* *X  
**  
**  
XX
```

Minimum distance: 2; Closest character: s
The character is a: s
Character name: s

```
***X  
X*  
****  
*X  
X****
```

Minimum distance: 18; Closest character: t
Unknown character.
Character name: ??

```
XX  
**  
X*XX*X  
**  
**  
*X  
**X
```

Minimum distance: 5; Closest character: u
The character is a: u
Character name: u

```
XX XX  
** **  
** **  
*X *X  
*** *X
```

Minimum distance: 0; Closest character: v
The character is a: v
Character name: v

```
XX XX  
** **  
*X X*  
*XX*  
**
```

Minimum distance: 0; Closest character: w
The character is a: w
Character name: w

```
XX XX  
** X **  
**XXX**  
**X**  
X* *X
```

Minimum distance: 0; Closest character: x
The character is a: x
Character name: x

```
X* *X
*XX*
XX
*XX*
X* *X
```

Minimum distance: 0; Closest character: "
The character is a: "
Character name: "

```
XX XX
** **
XX XX
```

Minimum distance: 0; Closest character: y
The character is a: y
Character name: y

```
XX XX
** **
*X **
*****
**
X***
```

Minimum distance: 0; Closest character: #
The character is a: #
Character name: #

```
XX XX
** **
XXX**XXX
XX XX
XXX**XXX
** **
XX XX
```

Minimum distance: 0; Closest character: z
The character is a: z
Character name: z

```
X****X
**
**
**
X****X
```

Minimum distance: 21; Closest character: +
Unknown character.
Character name: ??

```
X
**X*X
X* *
**X**
* *X
X*X**
X
```

Minimum distance: 4; Closest character: !
The character is a: !
Character name: !

```
XX
**
**
**
XX
XX
```

Minimum distance: 0; Closest character: %
The character is a: %
Character name: %

```
XX X
XX **
**
**
**
** XX
X XX
```

Minimum distance: 10; Closest character: &
The character is a: &
Character name: &

```
****
*X X*
*X X*
****
*X X*X
*X *X
*****X
```

Minimum distance: 4; Closest character: *
The character is a: *
Character name: *

```
X* *X
****
X*X**X*X
****
X* *X
```

Minimum distance: 12; Closest character: l
The character is a: l
Character name: l

```
XX
**
XX
```

Minimum distance: 0; Closest character: +
The character is a: +
Character name: +

```
XX
**
X*XX*X
**
XX
```

Minimum distance: 0; Closest character: (
The character is a: (
Character name: (

```
*X
**
*X
**
*X
**
*X
```

Minimum distance: 0; Closest character: ,
The character is a: ,
Character name: ,

```
XX
X*
X*
```

Minimum distance: 0; Closest character:)
The character is a:)
Character name:)

```
X*
**
X*
**
X*
**
X*
```

Minimum distance: 0; Closest character: -
The character is a: -
Character name: -

```
X****X
```

Minimum distance: 0; Closest character: .
The character is a: .
Character name: .

```
XX
XX
```

Minimum distance: 0; Closest character: 2
The character is a: 2
Character name: 2

```
****
X* X*
  X*
  **
  **
  **
X****X
```

Minimum distance: 0; Closest character: /
The character is a: /
Character name: /

```
*X
**
**
**
**
**
X*
```

Minimum distance: 0; Closest character: 3
The character is a: 3
Character name: 3

```
****
X* X*
  X*
  X**
  X*
X* X*
****
```

Minimum distance: 4; Closest character: Q
The character is a: Q
Character name: Q

```
****
*X X*
** **
** *X*
*X* **
** X*
****
```

Minimum distance: 0; Closest character: 4
The character is a: 4
Character name: 4

```
*X
***
****
X* X*
X***XXX
**
XX
```

Minimum distance: 0; Closest character: 1
The character is a: 1
Character name: 1

```
XX
**
XX*
**
**
**
**
X****X
```

Minimum distance: 0; Closest character: 5
The character is a: 5
Character name: 5

```
X****X
*X
X****
  X*
  **
X* X*
****
```

Minimum distance: 0; Closest character: 6
The character is a: 6
Character name: 6

```
****
*X *X
**
*X***
** X*
*X X*
****
```

Minimum distance: 0; Closest character: :
The character is a: :
Character name: :

```
XX

XX
```

Minimum distance: 0; Closest character: 7
The character is a: 7
Character name: 7

```
X****X
X* **
**
*X
**
**
XX
```

Minimum distance: 0; Closest character: ;
The character is a: ;
Character name: ;

```
XX

XX
X*
X*
```

Minimum distance: 0; Closest character: 8
The character is a: 8
Character name: 8

```
****
*X X*
*X X*
****
*X X*
*X X*
****
```

Minimum distance: 0; Closest character: <
The character is a: <
Character name: <

```
**X
**
**
X*
**
**
**X
```

Minimum distance: 0; Closest character: 9
The character is a: 9
Character name: 9

```
****
*X X*
*X **
***X*
**
X* X*
****
```

Minimum distance: 0; Closest character: =
The character is a: =
Character name: =

```
X****X

X****X
```

Minimum distance: 0; Closest character: >
The character is a: >
Character name: >

```
X**
**
**
 *X
**
**
X**
```

Minimum distance: 0; Closest character: C
The character is a: C
Character name: C

```
****
*X *X
**
**
**
*X *X
****
```

Minimum distance: 0; Closest character: ?
The character is a: ?
Character name: ?

```
****
X* X*
 X*
**
X*
XX
```

Minimum distance: 4; Closest character: D
The character is a: D
Character name: D

```
X*****
** **
** X*
** **
** X*
** **
X*****
```

Minimum distance: 0; Closest character: A
The character is a: A
Character name: A

```
**
*XX*
** **
*X**X*
** **
** **
XX XX
```

Minimum distance: 8; Closest character: E
The character is a: E
Character name: E

```
X*****X
**
**
*X*X
**
**
X*****X
```

Minimum distance: 10; Closest character: B
The character is a: B
Character name: B

```
X*****
** X*
** X*
*X***
** X*
** X*
X*****
```

Minimum distance: 8; Closest character: 7
The character is a: 7
Character name: 7

```
X*****X
**
**
*X*X
**
**
XX
```

Minimum distance: 4; Closest character: G
The character is a: G
Character name: G

```
*****
*X *X
**
** X*X
** **
*X **
****X
```

Minimum distance: 2; Closest character: K
The character is a: K
Character name: K

```
XX *X
** **
****
**X
****
** **
XX *X
```

Minimum distance: 0; Closest character: H
The character is a: H
Character name: H

```
XX XX
** **
** **
*X**X*
** **
** **
XX XX
```

Minimum distance: 0; Closest character: L
The character is a: L
Character name: L

```
XX
**
**
**
**
**
X****X
```

Minimum distance: 0; Closest character: I
The character is a: I
Character name: I

```
X**X
**
**
**
**
**
X**X
```

Minimum distance: 0; Closest character: M
The character is a: M
Character name: M

```
X* *X
*** **
*****
** X **
** **
** **
XX XX
```

Minimum distance: 0; Closest character: J
The character is a: J
Character name: J

```
X**X
**
**
**
**
X* X*
***
```

Minimum distance: 4; Closest character: N
The character is a: N
Character name: N

```
X* XX
*** **
***X**
**X***
** **
** **
XX XX
```

Minimum distance: 0; Closest character: O
The character is a: O
Character name: O

```
*****  
*X X*  
** **  
** **  
** **  
** **  
*X X*  
*****
```

Minimum distance: 0; Closest character: S
The character is a: S
Character name: S

```
*****  
*X *X  
*X  
*****  
X*  
X* X*  
*****
```

Minimum distance: 8; Closest character: P
The character is a: P
Character name: P

```
X*****  
** X*  
** X*  
*X***  
**  
**  
X*X
```

Minimum distance: 0; Closest character: T
The character is a: T
Character name: T

```
X*****X  
**  
**  
**  
**  
**  
**  
XX
```

Minimum distance: 2; Closest character: Q
The character is a: Q
Character name: Q

```
*****  
*X X*  
** **  
** **  
** X*  
*X **  
** *X
```

Minimum distance: 0; Closest character: U
The character is a: U
Character name: U

```
XX XX  
** **  
** **  
** **  
** **  
*X X*  
*****
```

Minimum distance: 12; Closest character: R
The character is a: R
Character name: R

```
X*****  
** X*  
** X*  
***X*  
**X*  
** **  
X*X *X
```

Minimum distance: 0; Closest character: V
The character is a: V
Character name: V

```
XX XX  
** **  
** **  
** **  
*X X*  
*XX*  
**
```

Minimum distance: 0; Closest character: W
The character is a: W
Character name: W

```
XX XX
** **
** **
** X **
*****
*** ***
X* *X
```

Minimum distance: 0; Closest character: X
The character is a: X
Character name: X

```
XX XX
*X X*
*XX*
XX
*XX*
*X X*
XX XX
```

Minimum distance: 0; Closest character: Y
The character is a: Y
Character name: Y

```
XX XX
** **
*X X*
*XX*
**
**
XX
```

Minimum distance: 0; Closest character: Z
The character is a: Z
Character name: Z

```
X****X
**
**
**
**
**
X****X
```

Full results for `Ult'.chr

Cognition phase...
Done learning characters.

(D)isplay char, (R)un new set, or (Q)uit: r

Filename of set to be run: ult.char

Minimum distance: 2; Closest character: @
The character is a: @
Character name: @

```
***
** **
*X ***
** X*X
**
*X *X
****
```

Minimum distance: 4; Closest character: 0
The character is a: 0
Character name: 0

```
***X
*X **
** **
*X *X
** *X
```

Minimum distance: 13; Closest character: b
Unknown character.
Character name: ??

```
XX
**
**
*X**
** X*
X* X*
X* **
```

Minimum distance: 4; Closest character: o
The character is a: o
Character name: o

```
****
*X *X
**
*X *X
****
```

Minimum distance: 0; Closest character: g
The character is a: g
Character name: g

```
****X
*X **
*X **
***X*
**
X****
```

Minimum distance: 13; Closest character: d
Unknown character.
Character name: ??

```
XX
**
**
**X*
*X **
*X *X
** *X
```

Minimum distance: 8; Closest character: h
The character is a: h
Character name: h

```
XX
**
**
*X**
** X*
** *X
XX *X
```

Minimum distance: 2; Closest character: e
The character is a: e
Character name: e

```
****
** **
*X***X
**
****X
```

Minimum distance: 2; Closest character: i
The character is a: i
Character name: i

```
XX
XX
**
**
X**X
```

Minimum distance: 16; Closest character: 6
Unknown character.
Character name: ??

```
***
*X *X
**
XXX**X
**
X*
X*
```

Minimum distance: 14; Closest character: 8
Unknown character.
Character name: ??

```
XX
XX
**
**
XX **
*X X*
****
```

Minimum distance: 8; Closest character: k
The character is a: k
Character name: k

```
XX
**
** *X
** **
*X**
** **
XX *X
```

Minimum distance: 0; Closest character: o
The character is a: o
Character name: o

```
****
*X X*
** **
*X X*
****
```

Minimum distance: 2; Closest character: l
The character is a: l
Character name: l

```
X*X
**
**
**
**
**
X**X
```

Minimum distance: 5; Closest character: p
The character is a: p
Character name: p

```
X****
** X*
** X*
*X***
*X
X*X
```

Minimum distance: 11; Closest character: k
The character is a: k
Character name: k

```
X X
** **
*X**X*
** **
XX XX
```

Minimum distance: 5; Closest character: q
The character is a: q
Character name: q

```
****X
*X **
*X X*
***X
*X
*X
```

Minimum distance: 4; Closest character: n
The character is a: n
Character name: n

```
X***
** X*
** **
** *X
XX *X
```

Minimum distance: 5; Closest character: r
The character is a: r
Character name: r

```
X* **
*X* *X
**
**
XX
```

Minimum distance: 2; Closest character: s
The character is a: s
Character name: s

```
***X
X*
****
 *X
X*****
```

Minimum distance: 12; Closest character: 8
The character is a: 8
Character name: 8

```
XX XX
** **
*X**X*
** **
X X
```

Minimum distance: 14; Closest character: 8
Unknown character.
Character name: ??

```
XX
**
XXX*X
**
**
*X *X
***
```

Minimum distance: 0; Closest character: x
The character is a: x
Character name: x

```
X* *X
*XX*
XX
*XX*
X* *X
```

Minimum distance: 7; Closest character: u
The character is a: u
Character name: u

```
XX XX
** **
** **
*X *X
** *X
```

Minimum distance: 2; Closest character: y
The character is a: y
Character name: y

```
XX XX
** **
*X **
*****
**
X**
```

Minimum distance: 0; Closest character: v
The character is a: v
Character name: v

```
XX XX
** **
*X X*
*XX*
**
```

Minimum distance: 19; Closest character: w
Unknown character.
Character name: ??

```
X****X
XX
X*XX*X
XX
X****X
```

Minimum distance: 6; Closest character: !
The character is a: !
Character name: !

XX
**
**
XX

XX
XX

Minimum distance: 0; Closest character: %
The character is a: %
Character name: %

XX X
XX **
**
**
**
** XX
X XX

Minimum distance: 12; Closest character: 5
The character is a: 5
Character name: 5

XX XX
X* X*
X* **

Minimum distance: 29; Closest character: w
Unknown character.
Character name: ??

X X
X X
*** *X
*X XXX
*X XXX
*** *X

Minimum distance: 22; Closest character: &
Unknown character.
Character name: ??

** **
XXX*XXX
** **
** **
XXX*XXX
** **

Minimum distance: 10; Closest character: l
The character is a: l
Character name: l

XX
X*
X*

Minimum distance: 20; Closest character: 8
Unknown character.
Character name: ??

X
***X
XXX

XXX
X***
X

Minimum distance: 0; Closest character: (
The character is a: (
Character name: (

*X
**
*X
**
*X
**
*X

Minimum distance: 0; Closest character:)
The character is a:)
Character name:)

X*
**
X*
**
X*
**
X*

Minimum distance: 0; Closest character: -
The character is a: -
Character name: -

X****X

Minimum distance: 4; Closest character: *
The character is a: *
Character name: *

X* *X

X*X**X*X

X* *X

Minimum distance: 0; Closest character: .
The character is a: .
Character name: .

XX
XX

Minimum distance: 0; Closest character: +
The character is a: +
Character name: +

XX
**
X*XX*X
**
XX

Minimum distance: 0; Closest character: /
The character is a: /
Character name: /

*X
**
**
**
**
X*

Minimum distance: 0; Closest character: ,
The character is a: ,
Character name: ,

XX
X*
X*

Minimum distance: 4; Closest character: O
The character is a: O
Character name: O

X X
** **
** **
** **
X X

Minimum distance: 9; Closest character: 1
The character is a: 1
Character name: 1

```
X
**
XX*
**
**
**
X**X
```

Minimum distance: 6; Closest character: 5
The character is a: 5
Character name: 5

```
X***X
*X
X***
  X*
  **
X* X*
***
```

Minimum distance: 6; Closest character: 2
The character is a: 2
Character name: 2

```
***
X* X*
  X*
  **
  **
  **
X***X
```

Minimum distance: 6; Closest character: J
The character is a: J
Character name: J

```
*X
**
**
****
** X*
*X X*
***
```

Minimum distance: 10; Closest character: 3
The character is a: 3
Character name: 3

```
***
X* X*
  X*
  X**
  X*
X* X*
***
```

Minimum distance: 8; Closest character: ?
The character is a: ?
Character name: ?

```
X***X
**
  X*
  **
  *X
  **
  XX
```

Minimum distance: 16; Closest character: V
Unknown character.
Character name: ??

```
XX XX
** **
** **
X**X*
**
**
XX
```

Minimum distance: 8; Closest character: 8
The character is a: 8
Character name: 8

```
***
*X X*
*X X*
***
*X X*
*X X*
***
```

Minimum distance: 4; Closest character: l
The character is a: l
Character name: l

```
***  
*X X*  
*X **  
****  
**  
**  
X*
```

Minimum distance: 4; Closest character: =
The character is a: =
Character name: =

```
X***X  
X***X
```

Minimum distance: 8; Closest character: +
The character is a: +
Character name: +

```
XX  
XX  
  
XX  
XX
```

Minimum distance: 4; Closest character: >
The character is a: >
Character name: >

```
X*  
**  
**  
*X  
**  
**  
X*
```

Minimum distance: 11; Closest character: ;
The character is a: ;
Character name: ;

```
XX  
XX  
  
XX  
**  
X
```

Minimum distance: 12; Closest character: ?
The character is a: ?
Character name: ?

```
***  
X* X*  
X*  
X*  
  
XX  
XX
```

Minimum distance: 4; Closest character: <
The character is a: <
Character name: <

```
*X  
**  
**  
X*  
**  
**  
*X
```

Minimum distance: 16; Closest character: R
Unknown character.
Character name: ??

```
**X  
****  
*X **  
** **  
*X*X*  
X* **  
X* XX
```

Minimum distance: 2; Closest character: B
The character is a: B
Character name: B

```
X*****  
** X*  
** X*  
*X**  
** X*  
** X*  
X*****
```

Minimum distance: 6; Closest character: F
The character is a: F
Character name: F

```
X*****X  
**  
**  
*X*X  
**  
X*  
X*
```

Minimum distance: 2; Closest character: C
The character is a: C
Character name: C

```
***  
** *X  
*X  
**  
**  
*X *X  
*****
```

Minimum distance: 10; Closest character: Q
The character is a: Q
Character name: Q

```
***  
** *X  
*X  
**  
** X*X  
*X **  
*****
```

Minimum distance: 4; Closest character: D
The character is a: D
Character name: D

```
X*****  
** X*  
** **  
** **  
** **  
** X*  
X*****
```

Minimum distance: 0; Closest character: H
The character is a: H
Character name: H

```
XX XX  
** **  
** **  
*X**X*  
** **  
** **  
XX XX
```

Minimum distance: 4; Closest character: E
The character is a: E
Character name: E

```
X*****X  
**  
**  
*X*X  
**  
**  
X*****X
```

Minimum distance: 0; Closest character: Z
The character is a: Z
Character name: Z

```
X*****X  
**  
**  
**  
**  
**  
X*****X
```

Minimum distance: 6; Closest character: U
The character is a: U
Character name: U

```
X***X
**
**
**
XX **
*X X*
****
```

Minimum distance: 8; Closest character: E
The character is a: E
Character name: E

```
X* XX
*** **
*** **
*****
** ***
** ***
XX *X
```

Minimum distance: 2; Closest character: K
The character is a: K
Character name: K

```
XX *X
** **
****
**X
****
** **
XX *X
```

Minimum distance: 0; Closest character: O
The character is a: O
Character name: O

```
****
*X X*
** **
** **
** **
*X X*
****
```

Minimum distance: 7; Closest character: L
The character is a: L
Character name: L

```
XX
**
**
**
**
X**
X* **X
```

Minimum distance: 2; Closest character: P
The character is a: P
Character name: P

```
X*****
** X*
** X*
*X***
**
**
X*X
```

Minimum distance: 4; Closest character: E
The character is a: E
Character name: E

```
X X
** **
*****
*****
** **
** **
XX XX
```

Minimum distance: 6; Closest character: O
The character is a: O
Character name: O

```
****
*X X*
** **
** **
** X*
*****
** *X
```

Minimum distance: 9; Closest character: H
The character is a: H
Character name: H

```
X*****  
** X*  
** X*  
*X**X  
** *X  
** X*  
XX XX
```

Minimum distance: 0; Closest character: V
The character is a: V
Character name: V

```
XX XX  
** **  
** **  
** **  
*X X*  
*XX*  
**
```

Minimum distance: 11; Closest character: =
The character is a: =
Character name: =

```
***  
** *X  
X*  
****  
X*  
X* X*  
****
```

Minimum distance: 12; Closest character: E
The character is a: E
Character name: E

```
XX XX  
** **  
** **  
** **  
*X**X*  
** **  
X X
```

Minimum distance: 0; Closest character: T
The character is a: T
Character name: T

```
X*****X  
**  
**  
**  
**  
**  
**  
XX
```

Minimum distance: 0; Closest character: X
The character is a: X
Character name: X

```
XX XX  
*X X*  
*XX*  
XX  
*XX*  
*X X*  
XX XX
```

Minimum distance: 0; Closest character: U
The character is a: U
Character name: U

```
XX XX  
** **  
** **  
** **  
** **  
*X X*  
****
```

Minimum distance: 0; Closest character: Y
The character is a: Y
Character name: Y

```
XX XX  
** **  
*X X*  
*XX*  
**  
**  
XX
```

Minimum distance: 14; Closest character: E
Unknown character.
Character name: ??

X****X
**
**
X*XX*X
**
**
X****X