

Chinese Typesetting

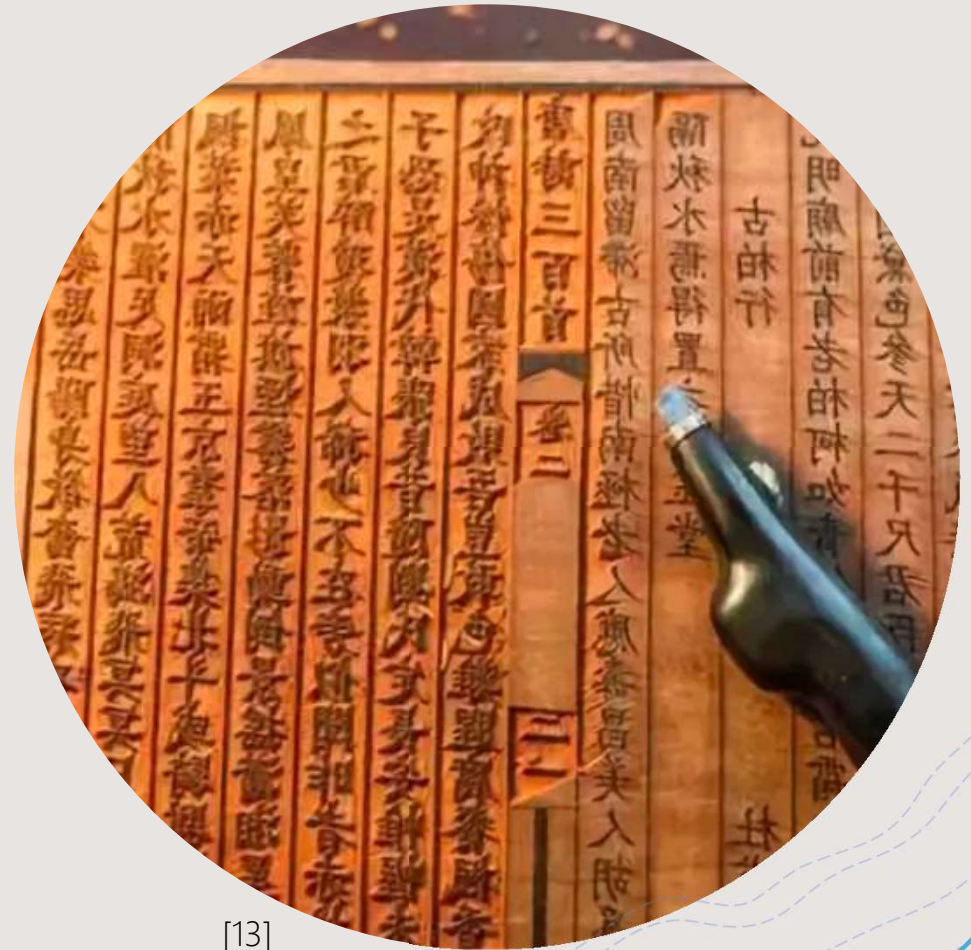
+

Laser Phototypesetting for Chinese Characters

Before Laser Phototypesetting

Woodblock Printing

- People engrave the mirrored characters on a whole woodblock, ink the woodblock, and press the paper on the woodblock to get a copy of the document.
- Calligraphers and Engravers determine the typesetting
- Engrave on a new woodblock if the articles get updated
- Woodblock is not durable so that characters on the woodblock will gradually deteriorate



[13]

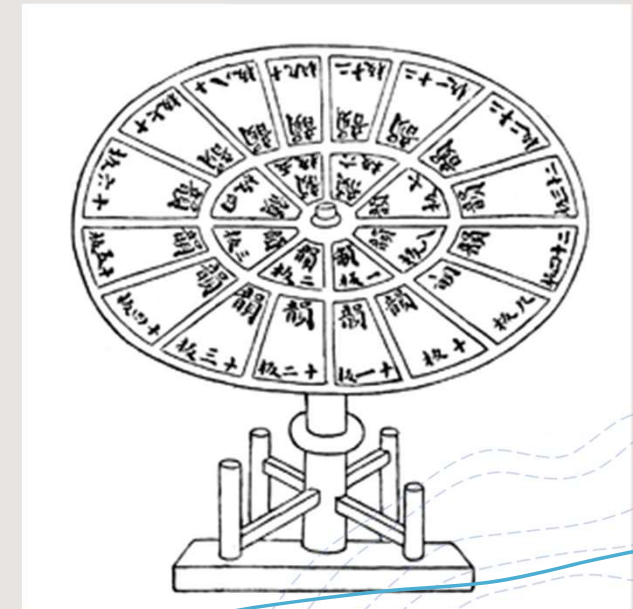
Before Laser Phototypesetting

Moveable Typesetting

- Engrave characters on clay, wood, or bronze types
- Invent a multi-area round table for sorting types
- Classify characters according to their pronunciation
- Each area of the table corresponds to one rhyme
- 张 (zhang) and 长 (chang) can lie in the same areas because their Pinyin both end with “ang”



[14]



[15]

Before Laser Phototypesetting

Woodblocks were more popular than moveable type before the 19th century. Why?

- Sorting and selecting moveable type need workers with a high level of literacy.
- Preparing moveable type for tens of thousands of characters was expensive and time-consuming. (e.g., Zhen Wang of Yuan Dynasty spent three years manufacturing moveable type for around 30,000 characters.)

Before Laser Phototypesetting

Hot Metal Typesetting & Film Typesetting

- Skilled operators achieve speeds of around 1000 characters/hour (vs. one woodblock engraver engraves about 200 characters per day)
- Thorough manipulation needs around 3 years of training
- Few aids for text storage, editing and composition

Electronic phototypesetting

- Must have resolution higher than 25 lines/mm (635 lines/inch)
- Need high-capacity storage to store typefaces and fonts
- Very expensive and not more cost-effective

An alcove storing hot-metal type



A flat-bed filmsetter

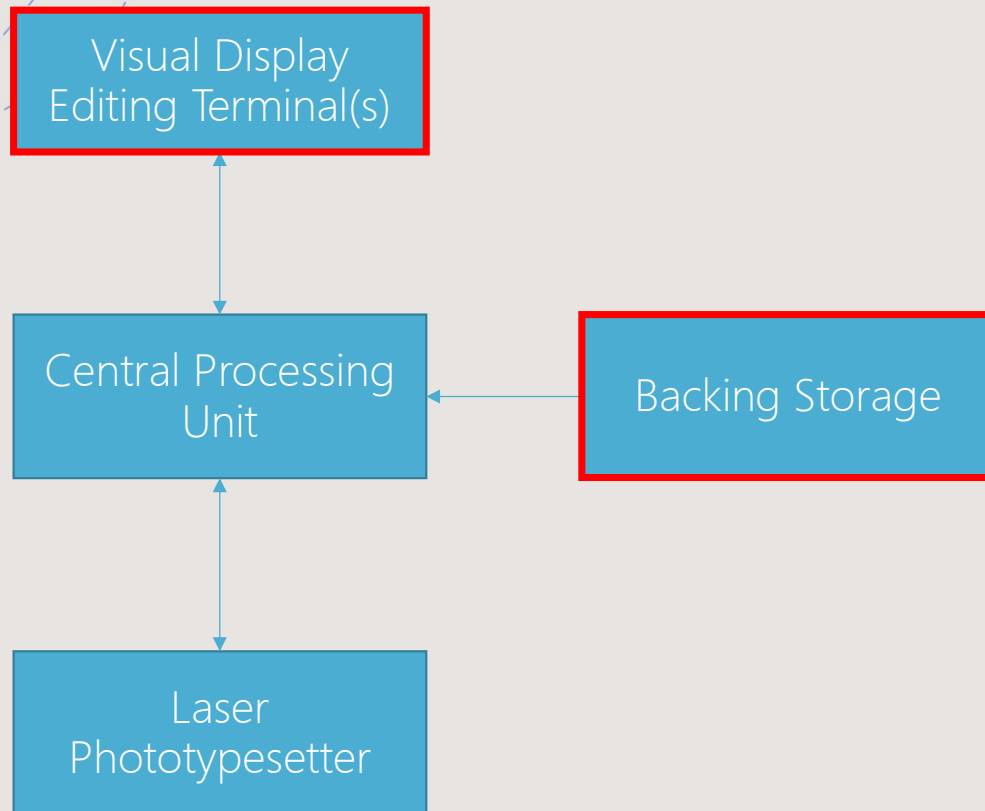


Laser Phototypesetting

- Necessary equipment became cheaper, and more techniques appeared.
- During the 1970s and the 1980s, developers invented many products of laser phototypesetting.
- Two typical laser phototypesetting systems :
 1. British Monotype's Lasercomp
 2. HuaGuang Laser phototypesetting system

British Monotype's Lasercomp

A fourth-generation laser phototypesetting system with digital storage



A visual-display editing terminal with a Loh keyboard

Loh Keyboard

Professor S C Loh developed this keyboard system at the Chinese University of HK

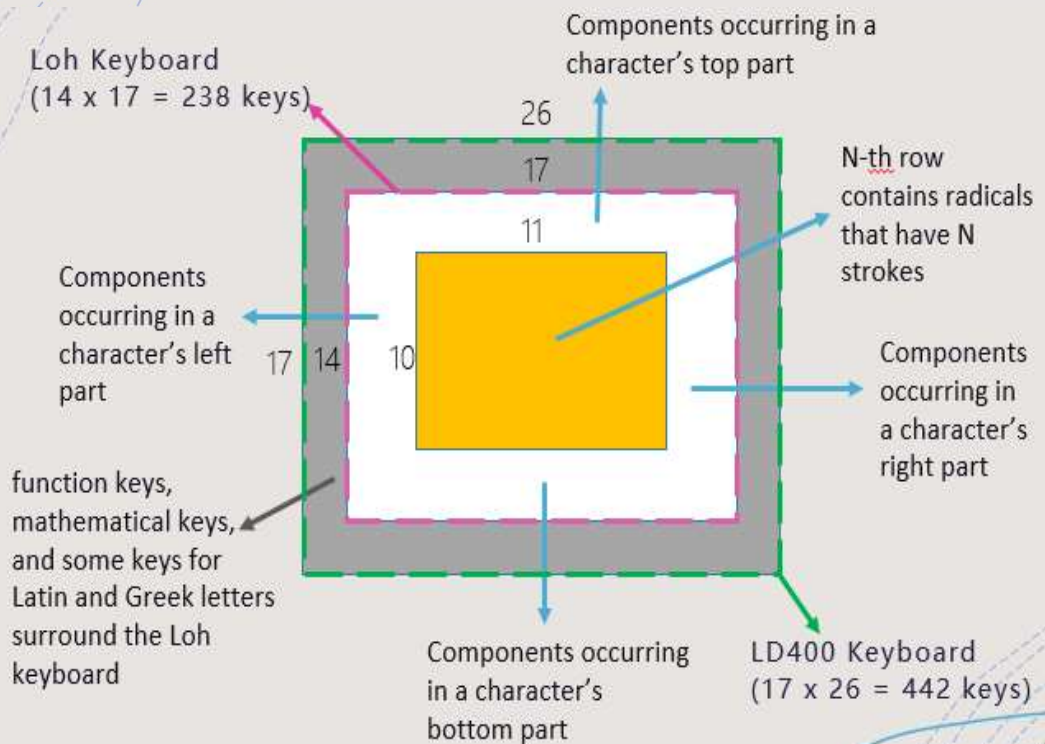
A Loh keyboard can represent over 13000 Chinese characters without using the two shift keys.

The average number of key depressions for each character in standard book work 2 ~ 4.

The whole laser phototypesetting system can include up to 10 keyboards.



The layout of the Loh Keyboard



Loh Keyboard

The sequence of keys representing a character derives from the stroke sequence that people use to draw this character.

Example: "read"



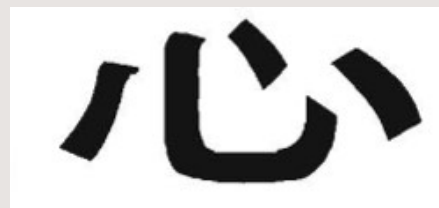
1



2

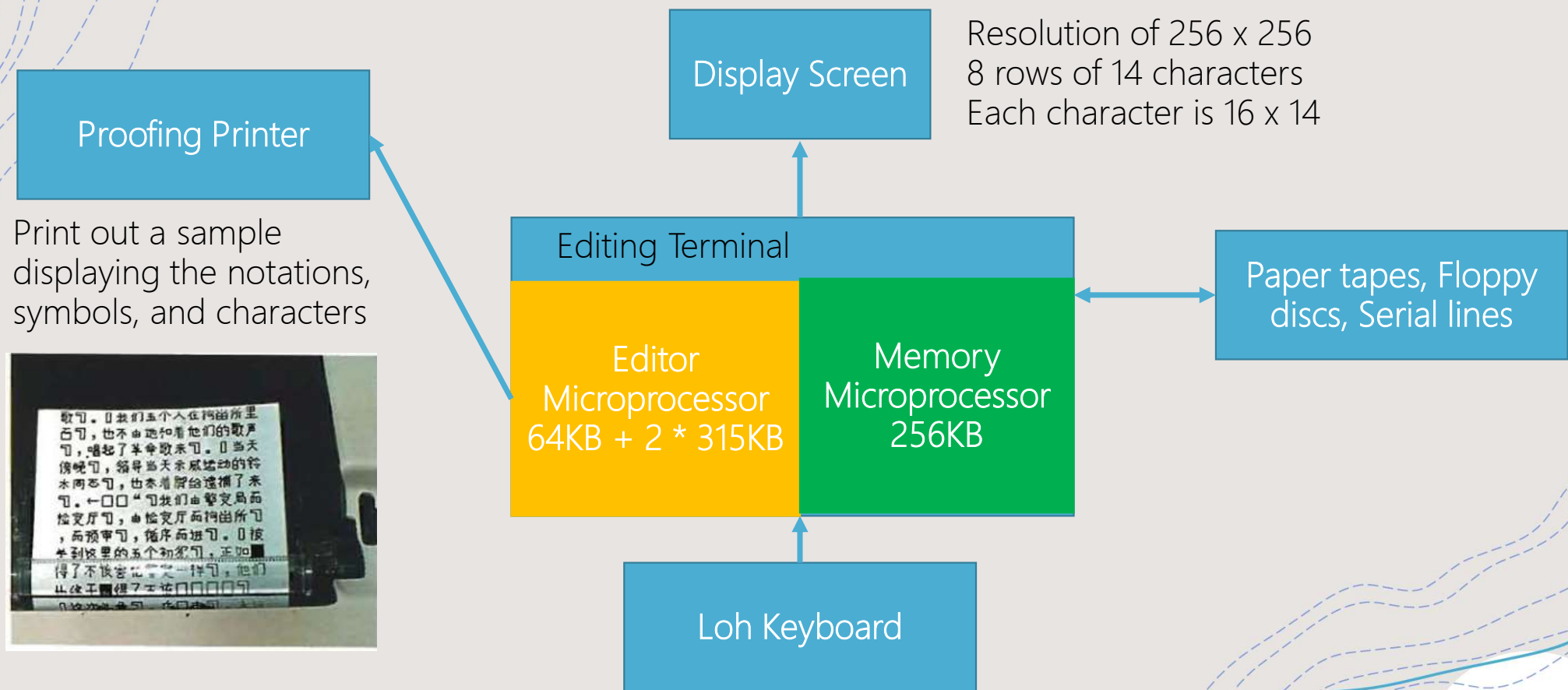


3



4

Visual Display Editing Terminal

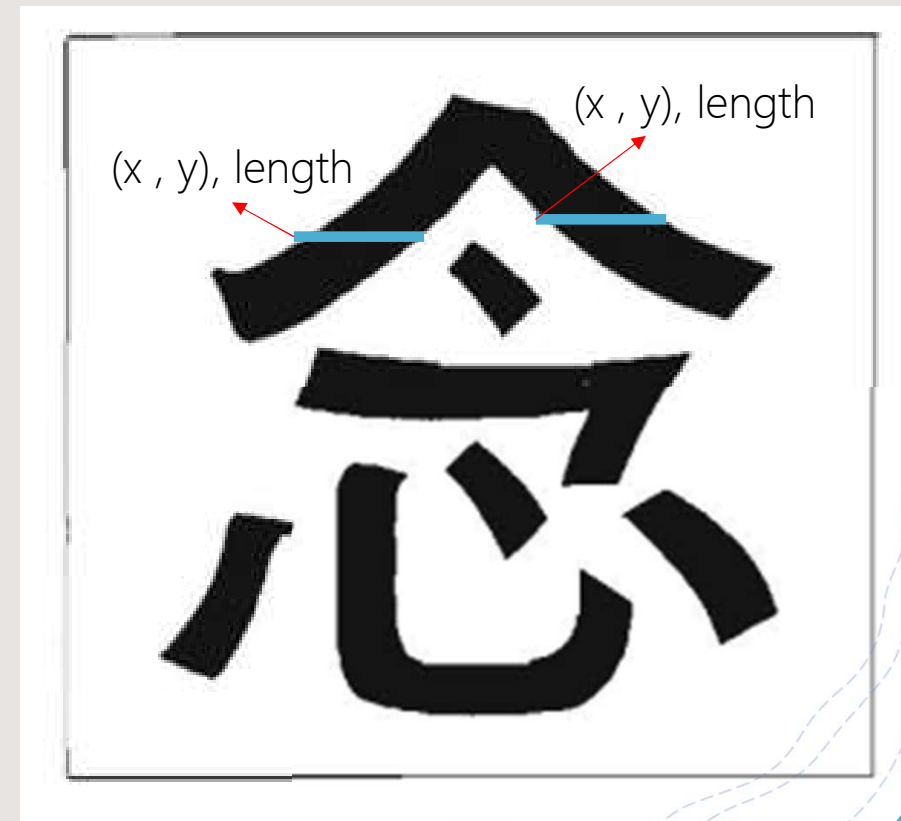


Backing Storage

- Scan each character at a resolution of 1000 lines/inch and store the bitmap of each character on the magnetic discs
- Memory capacity ranges from 80MB to 320MB or more (at least around 60000 characters)
- Lasercomp uses the run-length coding technique to compress characters

Run-length coding technique

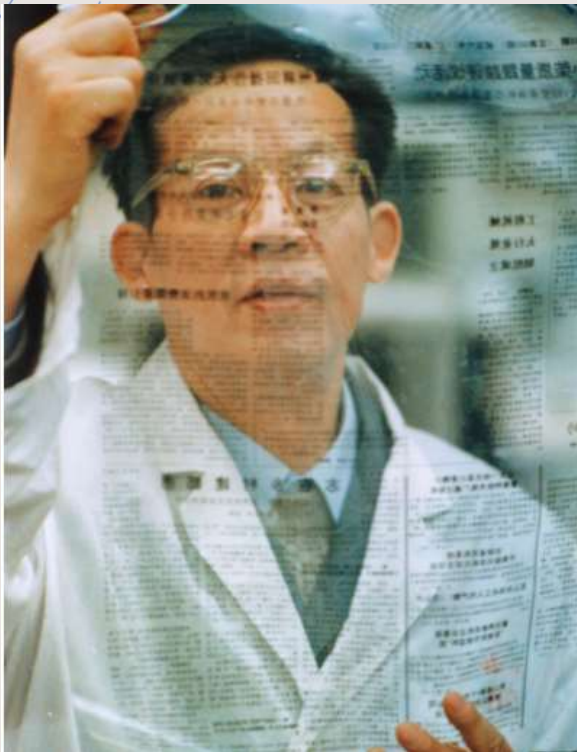
- Widely used in phototypesetting systems (e.g., Digisit (CRT typesetter))
- Record the *beginning* and the *length* of each separate black section
- Reduce the required storage by about 50% on average.



Drawbacks of Run-length coding technique

- We can't use a single representation to generate different sizes of a character
- Store different sizes of the same font in separate areas of a disc
- The low access speeds of removable discs become a bottleneck
- The storage supplies the character images at around 25 characters/ second

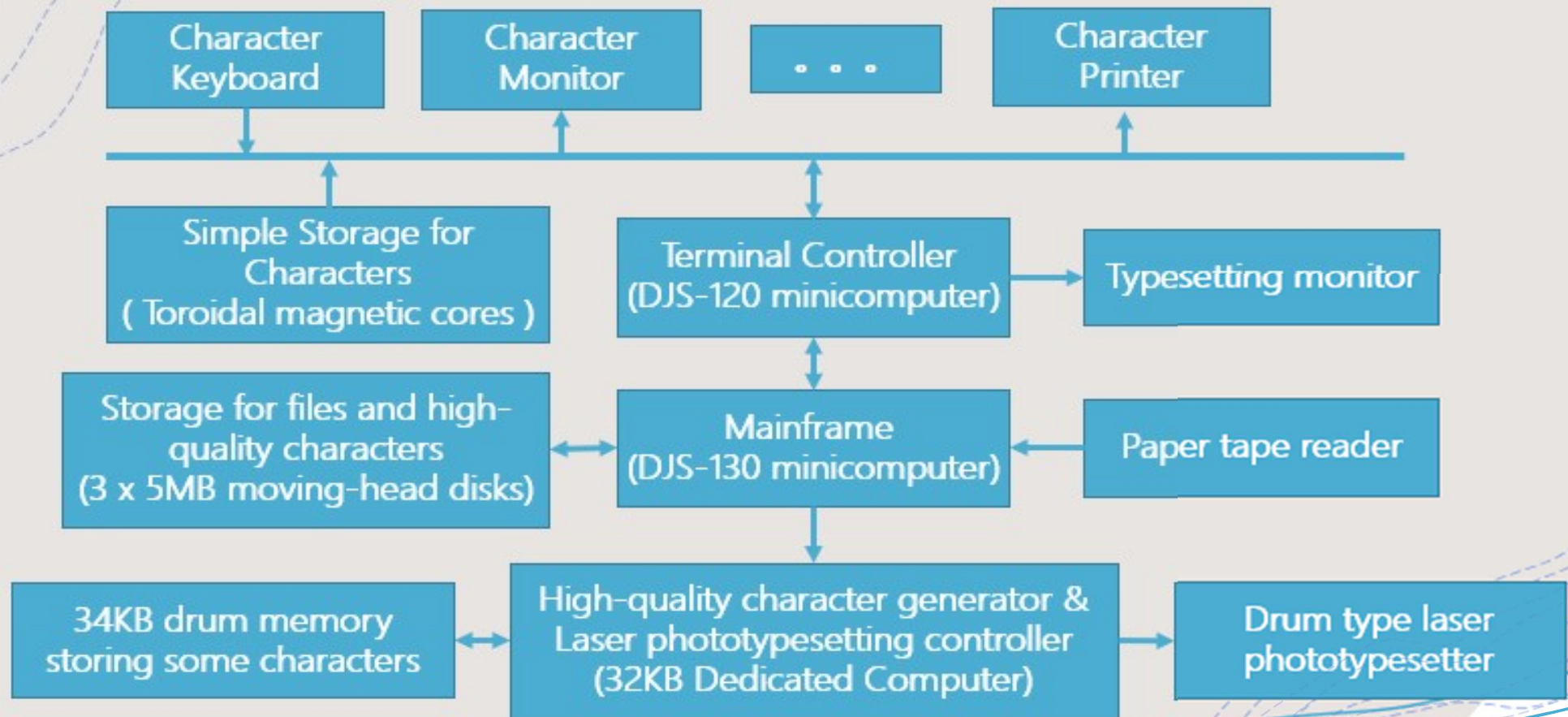
HuaGuang Laser Phototypesetting System



[17]

- Since 1974, Professor Xuan Wang from Peking University started developing his “***Computerized Chinese Character Editing and Laser Typesetting System***” .
- Initially called “**Hua Guang**” (We will denote this system as **the HG system** for short)
- The first proof-of-concept prototype got verified in 1981, and the whole development was finished in 1989.

Workflow of the HG system



HuaGuang BD Typesetting Language

- The typesetting language of the HG system consists of typesetting notations
- Each typesetting notation is denoted by an abbreviation containing several Pinyin initials.
- (e.g., KG = Kong Ge = Whitespace, YM = Ye Ma = Page , etc.)
- Operators embed the typesetting notations into the initial articles to indicate fonts and locations of the characters
- Potential functions: start a new line, start a new page, mix Chinese and Western symbols, automatic page number, etc.

Example – HT command

- + An HT command sets a font and a size for characters following it
- + HT[size][font] (e.g., HT3BS = size is 3 and font is Bao Song)

[font]	Actual Font	[font]	Actual Font
BS	Bao Song	F	Imitation Song
H	Heiti	K	Kaiti
SS	Shu Song	XBS	Xiao Biao Song (Small Standard Song)
ZDX	Zhong Deng Xian (Mid Arial)	XDX	Xi Deng Xian (Thin Arial)

[size]	Actual size in mm
2	7.397
2"	6.369
3	5.547
4	4.931
4"	4.246
5	3.698
...	...

Example – HT command

- + If you give no arguments, you set the characters following the notation to the default size and font.
- + If you give a single number for the first argument, each character following the notation is squared.

报^空宋：〔 HT3BS 〕君问归期未有期
✓〔 HT 〕书^空宋：〔 HT3SS 〕巴山夜雨涨
秋池✓〔 HT 〕仿^空宋：〔 HT3F 〕何当共剪
西窗烛✓〔 HT 〕楷^空体：〔 HT3K 〕却话巴
山夜雨时✓〔 HT 〕黑^空体：〔 HT3H 〕独怜
幽草润边生✓〔 HT 〕小标宋：〔 HT3XBS 〕
上有黄鹂深树鸣〔 HT 〕



报 宋：君问归期未有期
书 宋：巴山夜雨涨秋池
仿 宋：何当共剪西窗烛
楷 体：却话巴山夜雨时
黑 体：独怜幽草润边生
小标宋：上有黄鹂深树鸣

Example – HT command

- + If you give two numbers separated by a comma for [size], the first number corresponds to the horizontal width and the second number corresponds to the height.

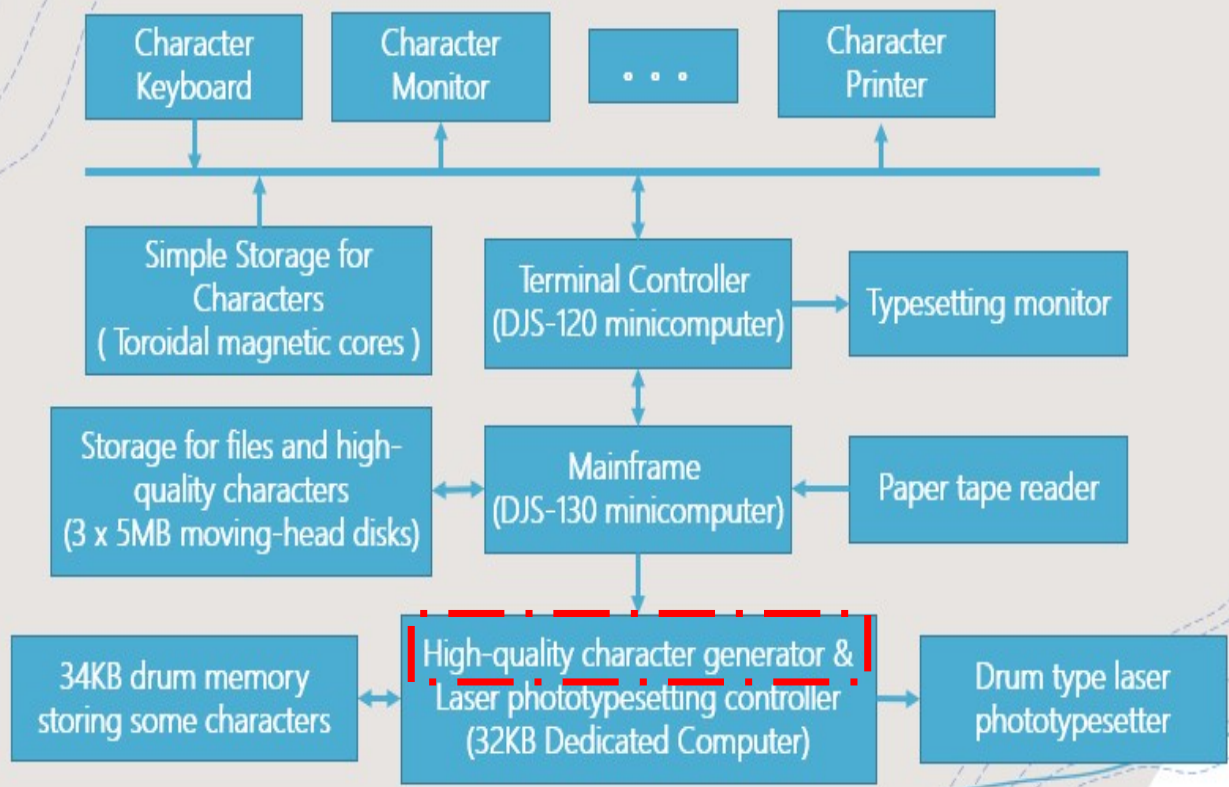
〔 HT2,4H 〕 排版系统✓

〔 HT5,2XBS 〕 生产日报表✓



排版系统
生产日报表

High-quality Chinese character generator



- An essential component that other competing products do not have.
- Store Chinese characters in compressed forms and convert these compressed forms into dot matrices

Storage requirements for high-resolution Chinese Characters

Character size		Dot matrix
in point	in mm ²	
56	19.66	576 x 576
48	16.86	494 x 494
42	14.74	432 x 432
36	12.63	370 x 370
31.5	11.06	324 x 324
28	9.83	288 x 288
24	7.37	216 x 216
18	6.28	184 x 184
15.75	5.53	162 x 162
14	4.91	144 x 144
12	4.23	124 x 124
10.5	3.68	108 x 108
9	3.14	92 x 92
7.875	2.76	82 x 82
7	2.46	72 x 72
6	2.12	62 x 62

Sizes of characters and their corresponding dot matrices.
The scanning resolution is 742 dpi in this table.

- + The massive number of Chinese characters is a crucial problem in developing a typesetting system
- + More than ten frequently used fonts (Song, long Song, boldface, long boldface, etc.). Each font has more than 8000 distinct characters.
- + A laser output device cannot change the dot size instantly
- + 10 fonts, each font has 16 sizes and 8000 characters. Each character of one size occupies $108 * 108$ bits on average, we need about $10 * 8000 * 16 * 108 * 108$ bits = 1.86624 GB

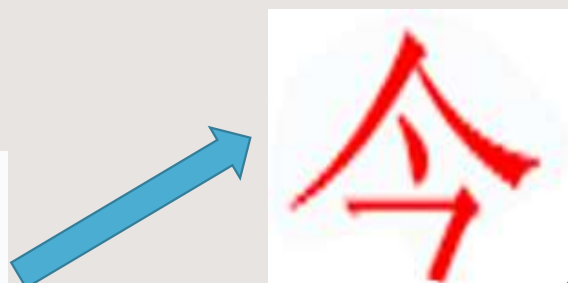
Divide characters into radicals and strokes

- Regard each character as a composition of radicals (and strokes) and only store radicals (and strokes) in the computer memory.

- Example: "read"



"read"



"today, now"



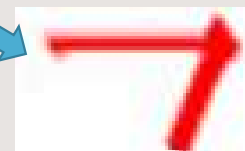
"heart" (radical & character)



"people"
(radical &
character)



Point stroke



Turn stroke

Divide characters into radicals and strokes

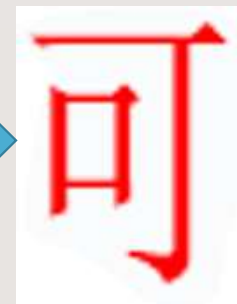
- Example: "River"



3-point water
radical (not a
character)



"River"

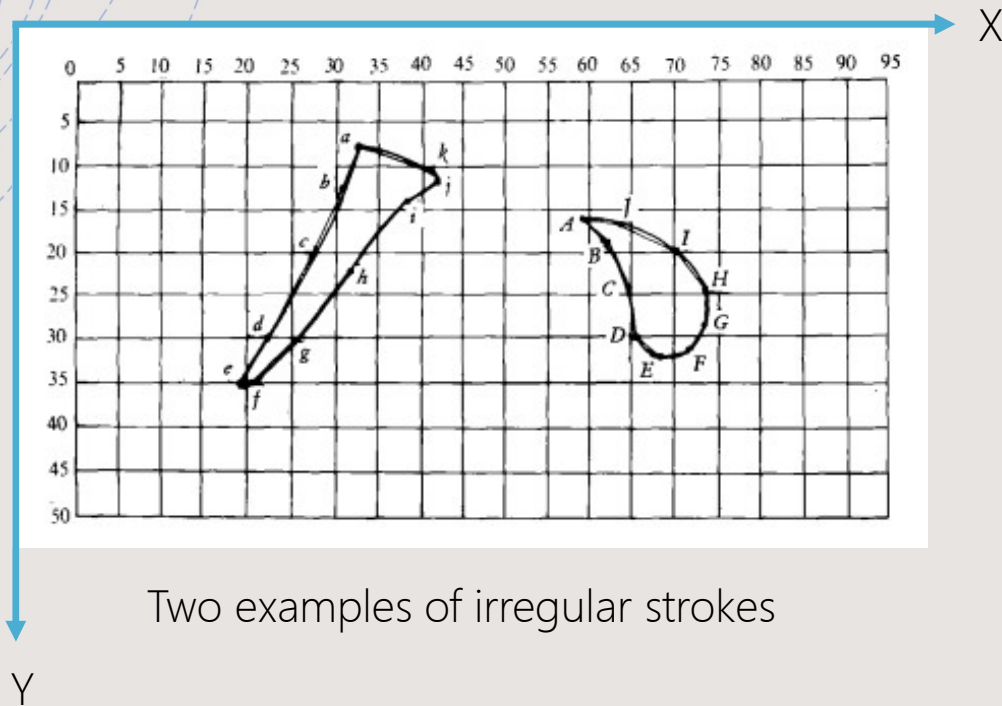


"can (do)"
(character)

Divide characters into radicals and strokes

- Generating one character includes generating a simpler character plus some radicals or strokes.
- About 700 radicals in total
- Select radicals from the computer memory and put them in their correct positions
- Good data compression ratio
- Reshaping a radical may cause degradation in the output quality of this character
- The HG system wants to reduce the storage requirement and remain a high output quality simultaneously
- If we want to represent radicals, we first need to know how to represent a stroke.
- Two kinds of strokes in characters: regular strokes and irregular strokes.

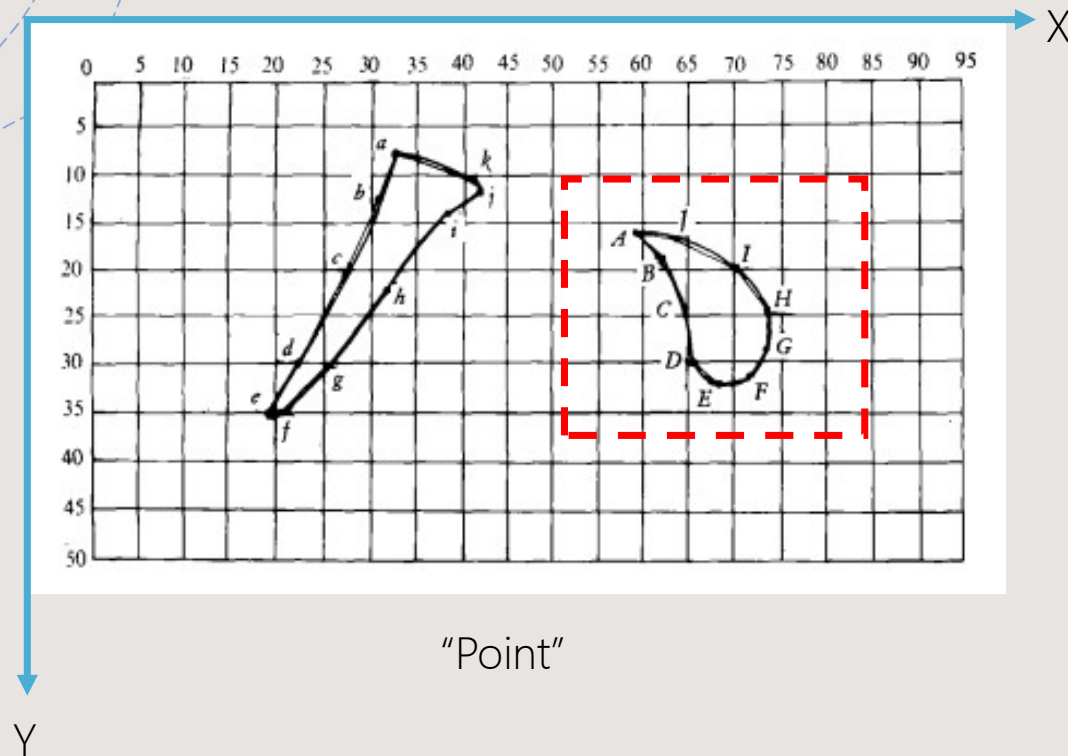
Compressed forms of irregular strokes



Two examples of irregular strokes

- We represent each irregular stroke by a series of signed vectors.
- The first vector indicates the beginning of the stroke. (X, Y)
- Each other vector indicates increments or decrements of X, Y coordinates. (ΔX , ΔY)

Example: "Point"



- Vector series:

A: [59, 15]

AB: [3, 3]

BC: [2, 5]

CD: [1, 5]

DE: [3, 2]

EF: [4, -1],

FG: [3, -3]

GH: [0, -3]

HI: [-3, -3]

IJ: [-7, -4]

JA: [-6, -1]

The character generator can automatically calculate the last vector to complete the contour, so we do not need to store the last vector.

Compress vector series

Case 1: $|X| < 16$ and $|Y| < 16$, and vectors lie in the same quadrant

7	6	5	4	3	2	1	0
0	0	Sign bit (0/1)	Sign bit (0/1)	N			
X				Y			
.....						

N vectors

Case 3: $16 \leq |X| < 64$ and $16 \leq |Y| < 64$

7	6	5	4	3	2	1	0
1	0	Sign bit (0/1)	Sign bit (0/1)	X		Y	
X				Y			

Case 2: $|X| < 8$ and $|Y| < 8$

7	6	5	4	3	2	1	0
0	1	Shared (0/1)	N				
Sign bit (0/1)	X			Sign bit (0/1)	Y		
.....						

N
vectors

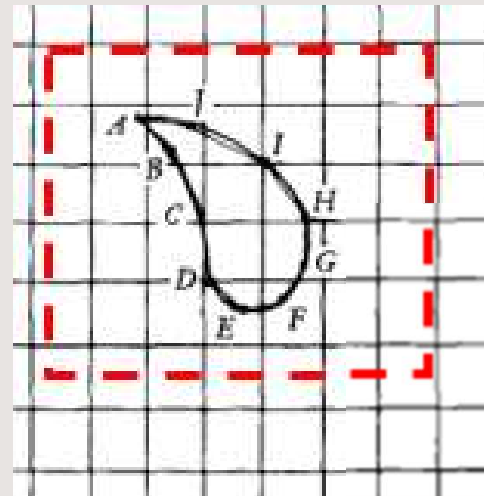
Beginning vector:

7	6	5	4	3	2	1	0
1	1	X					
X	Y						

The 5th highest control bit indicates whether the vector series is shared by multiple characters. Shared vector series are in a BMBD file.

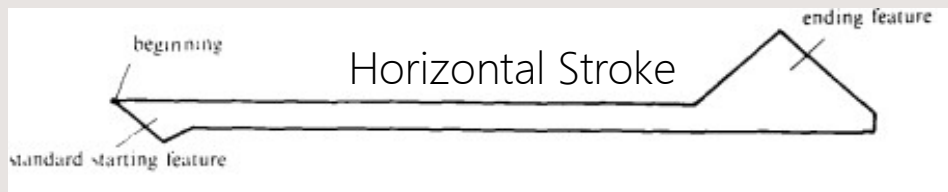
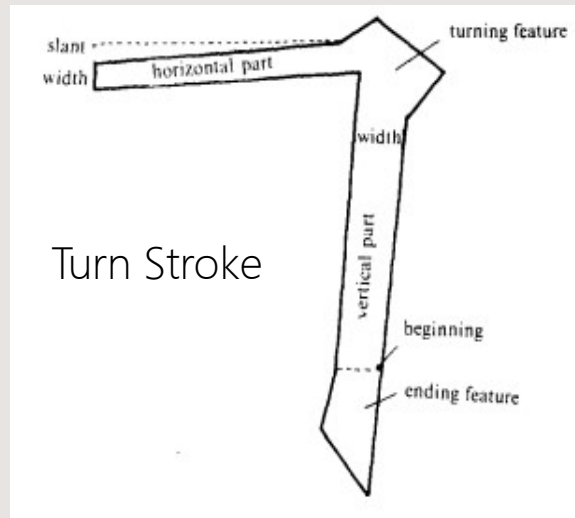
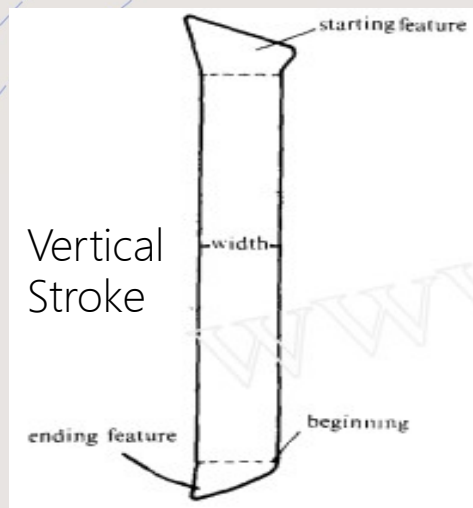
Why do we compress vector series?

- + Statistical studies about irregular strokes show that irregular strokes use short vectors more often than the long vectors.
- + Also, these short vectors change directions frequently.
- + In the example of the “Point” stroke, you will find that all vectors except the beginning vector have $|X| < 8$ and $|Y| < 8$.
- + The above method reduces the space occupied by vector series via making the short vectors more compact.



A: [59, 15]
AB: [3, 3]
BC: [2, 5]
CD: [1, 5]
DE: [3, 2]
EF: [4, -1],
FG: [3, -3]
GH: [0, -3]
HI: [-3, -3]
IJ: [-7, -4]
JA: [-6, -1]

Compressed forms of regular strokes

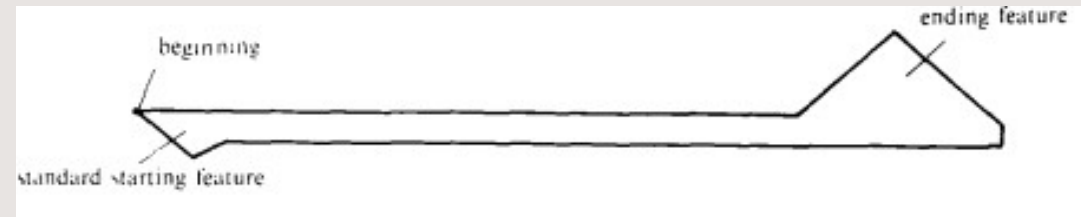


- The regular strokes occupy around 50% of all strokes in characters. They include horizontal, vertical, and turn strokes.
- Each regular stroke may have a starting feature, straight segments, turning features, and ending features.
- We refer to each feature by an ordinal number which points to an index in the BMBD file.
- We represent each feature by a sequence of vectors whose coordinates' absolute values are less than 8.
- We can represent each regular stroke by 3 to 6 bytes.

Example: Horizontal Stroke

- (a) A pair of beginning coordinates (X, Y)
- (b) Length
- (c) Ordinal number of the ending feature (2 to 4 bits)
- (d) Optional width variation (1 to 2 bits)
- (e) One optional bit indicating whether there is a standard starting feature

We need 3 to 4 bytes to represent a horizontal stroke of Song font.

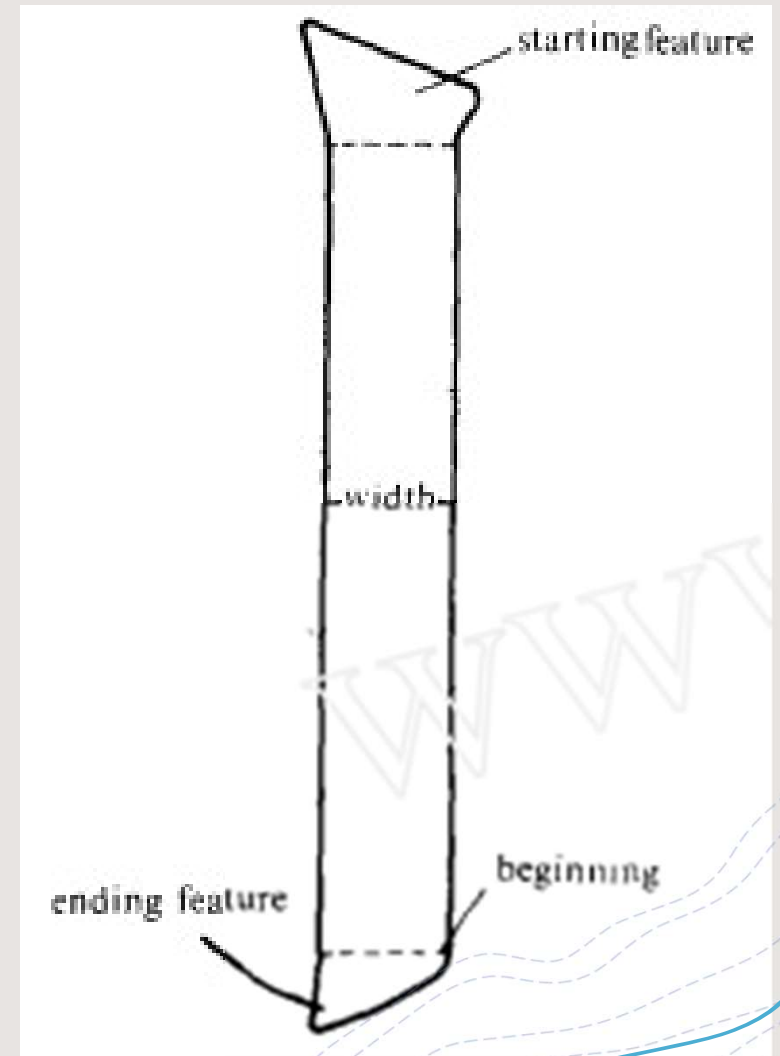


A horizontal stroke in the imitation Song typeface.

Example: Vertical Stroke

- (a) A pair of beginning coordinates (X, Y)
- (b) Length
- (c) Width (2 to 4 bits)
- (d) Ordinal number of the starting feature (2 to 5 bits)
- (e) Ordinal number of the ending feature (less than 4 bits)

We need 4 to 5 bytes to represent a vertical stroke of Song font.



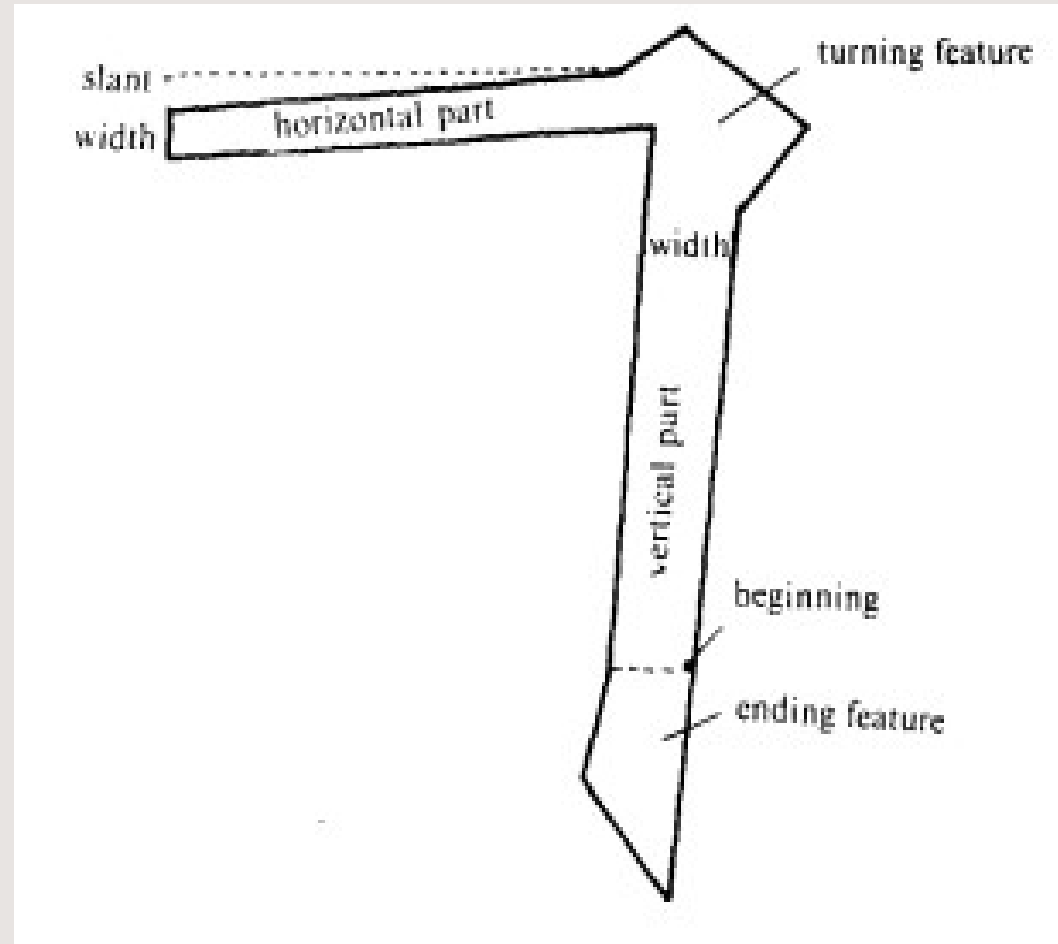
A vertical stroke in the imitation Song typeface.

Example: Turn Stroke

A pair of beginning coordinates (X, Y)

- (a) Vertical segment's length
- (b) Horizontal segment's length
- (c) Vertical segment's width
- (d) Horizontal segment's width
- (e) Ordinal number of the ending feature (2 to 4 bits)
- (f) Ordinal number of the turning feature (3 to 5 bits)
- (g) Indicator for whether the vertical segment is slanted (and its slant degree) (2 to 4 bites)
- (h) Slant degree of the horizontal part (1 to 3 bits)

We need 5 to 6 bytes to represent a turn stroke of imitation Song font.



A turn stroke in the imitation Song typeface.

Compressed forms of regular strokes

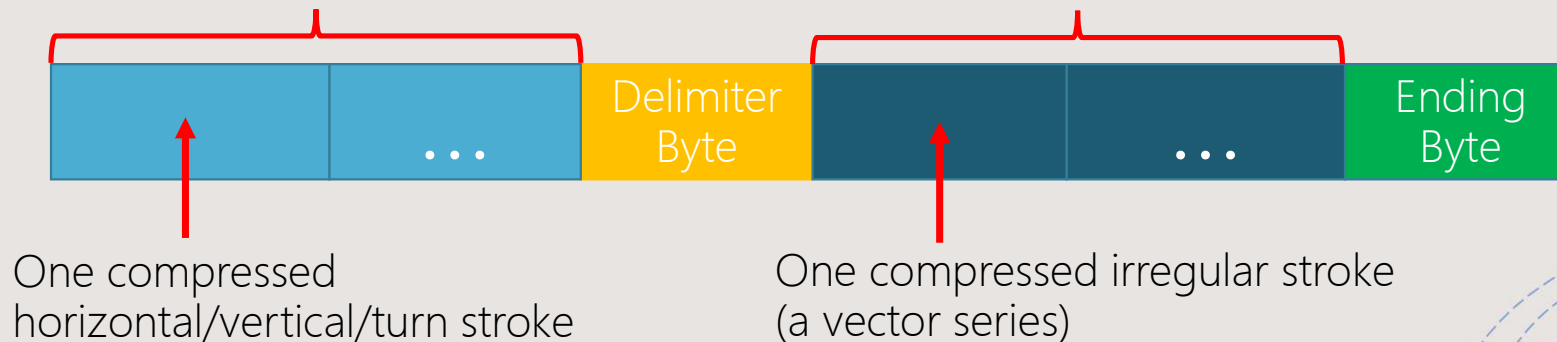
- Why don't we use vectors to contour a regular stroke?
- (1) Avoid long vectors whose coordinates' absolute values are larger than or equal to 16. We may need many vectors occupying two bytes or more.
- (2) We should avoid multiple copies of the same feature's vector series.
- (3) Due to rounding errors, distortion may occur when we scale up or down the stroke.

Compressed forms of Characters

- + A compressed character is essentially a sequence of compressed strokes.
- + Different font files store compressed characters of different fonts.
- + A table in the font file maps the ordinal number of a character to the number of bytes that this character's compressed version occupies.
- + We calculate the sum of the lengths of the compressed characters before the compressed character we want.

Compressed regular strokes

Compressed irregular strokes

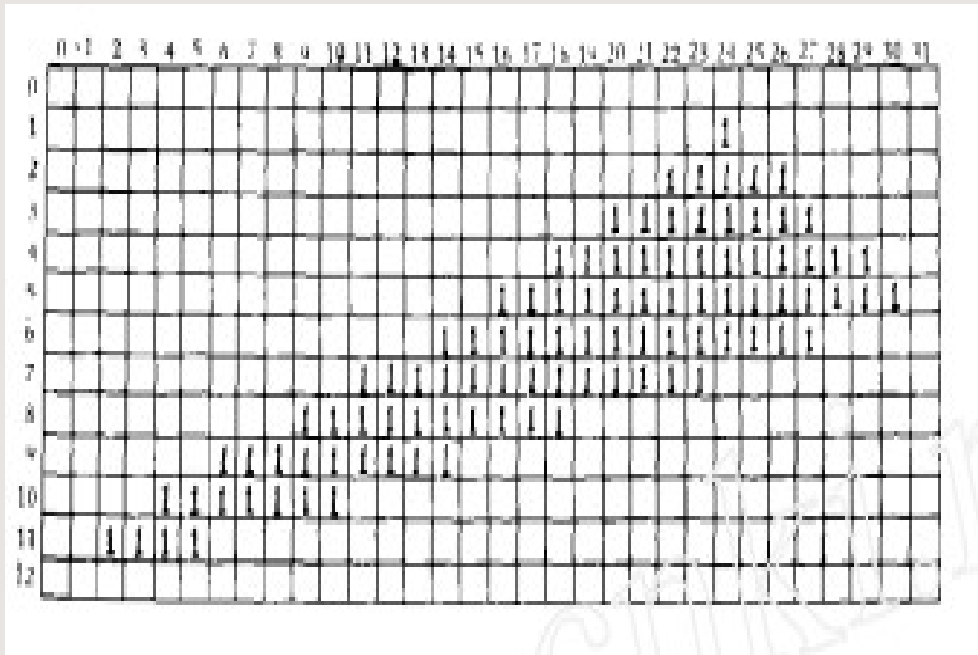


Accuracy of the compression

- The developers tested this method over a set of 7000 different characters with the Song typeface.
- Storage needed by a 10.5-point character decreases to 100 bytes on average.
- In tests, developers found that the quality did not get distorted or degraded.
- In this method, characters are no longer digital images but collections of numbers.

Decompress the compressed characters

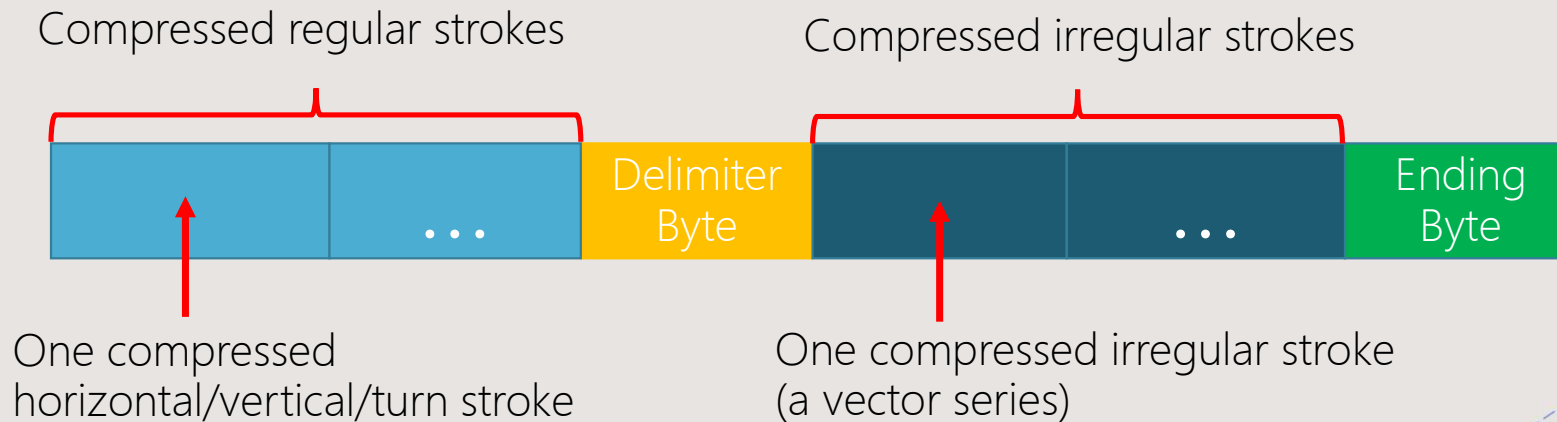
- + The input data to the high-quality character generator is the compressed representation of a character, and the output data is a dot matrix.



Final dot matrix of the stroke "Throw"

Step 1: Transform strokes to vectors

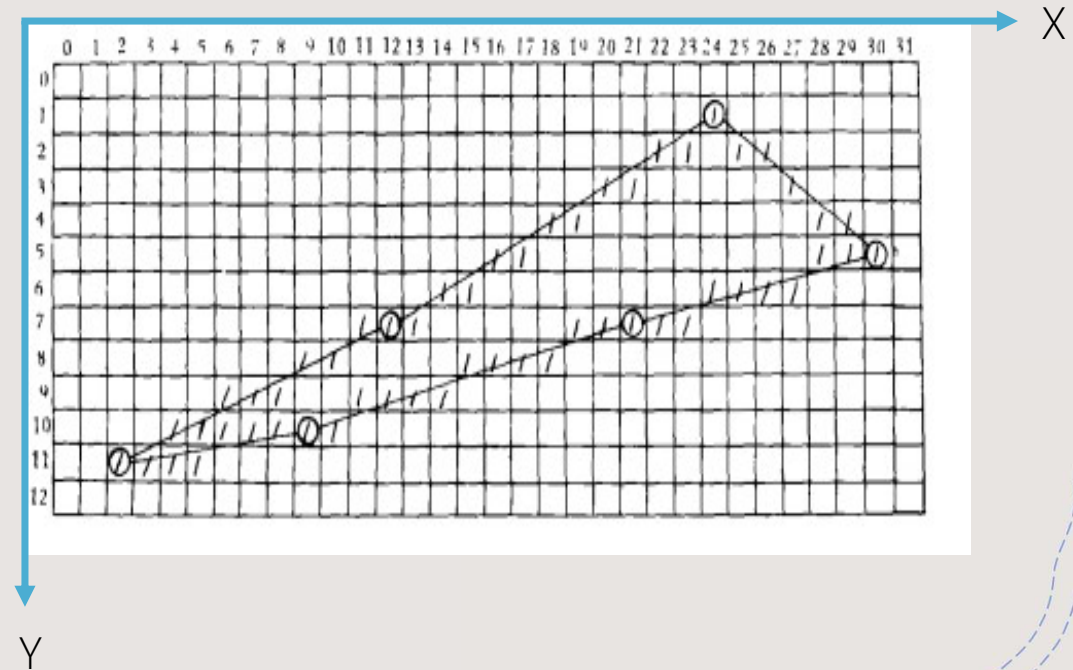
- + Given a compressed character, we convert every regular or irregular stroke to a sequence of vectors.
- + Each number in the vector is an 8-bit signed number.
- + Each vector now occupies two bytes



Step 2: Construct a Marked Dot Matrix

```
1 def sign(x):
2     if x < 0:
3         return -1
4     elif x == 0:
5         return 0
6     else:
7         return 1
8
9 # Here is a vector from (X, Y) to (X + deltaX, Y + deltaY)
10 # M is a grid (96 x 96)
11 def nearestDots(M, X, Y, deltaX, deltaY):
12     acc = 0
13     i = X
14     j = Y
15     while i != X + deltaX:
16         i += sign(deltaX)
17         acc += abs(deltaY)
18         if acc >= abs(deltaX)/2:
19             j += sign(deltaY)
20             M[i, j] = 1
21             acc -= abs(deltaX)
22         else:
23             M[i, j] = 1
24
```

An algorithm that finds nearest dots for a vector from (X, Y) to $(X + \Delta X, Y + \Delta Y)$.



The contoured dot matrix of an irregular stroke called **“Throw”**.

Step 2: Construct a Marked Dot Matrix

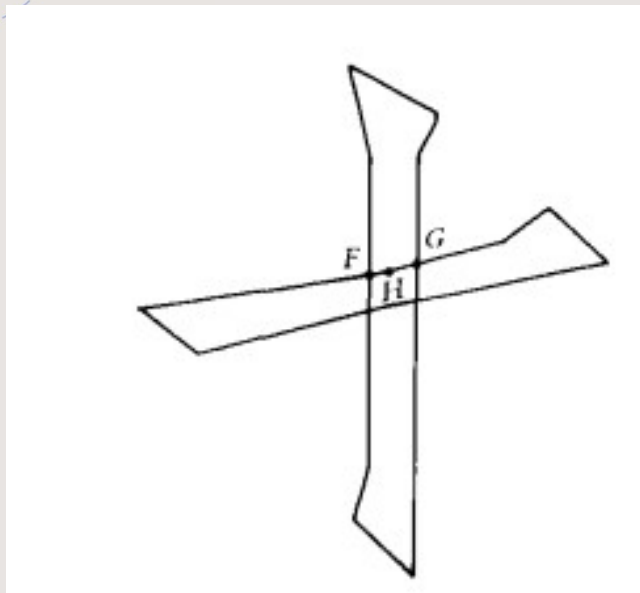
- + In a marked dot matrix, every dot position contains two bits. Every pair of two bits indicate the situation on that dot position.
 - + A dot on the left rim of the stroke gets a B-mark, and a dot on the right rim gets an E-mark. A prominent dot gets an I-mark.
- (1) 01 represents the beginning of a black section.
 - (2) 10 represents the end of a black section.
 - (3) 11 represents a single dot.
 - (4) 00 represents vacancy.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0																																
1																								11								
2																							01					10				
3																						01							10			
4																						01									10	
5																						01										10
6																						01										10
7																							01									
8																							01									
9																								01								
10																									01							
11																										01						
12																											01					

Marked Dot Matrix

Step 2: Construct a Marked Dot Matrix

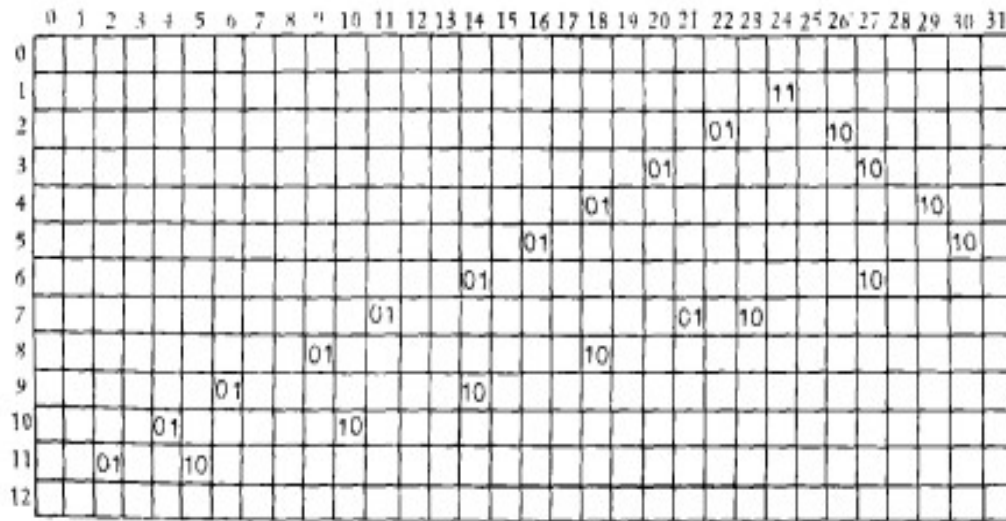
- + Strokes may intersect with each other
- + Consider the previously written 2-bit mark and the 2-bit mark to be written



Old mark	Mark to be written	Final mark
V	V	V
V	B	B
V	E	E
V	I	I
B	V	B
B	B	B
B	E	I
B	I	B
E	V	E
E	B	I
E	E	E
E	I	E
I	V	I
I	B	B
I	E	E
I	I	I

Step 3: Transform the Marked Dot Matrix to the Final Dot Matrix

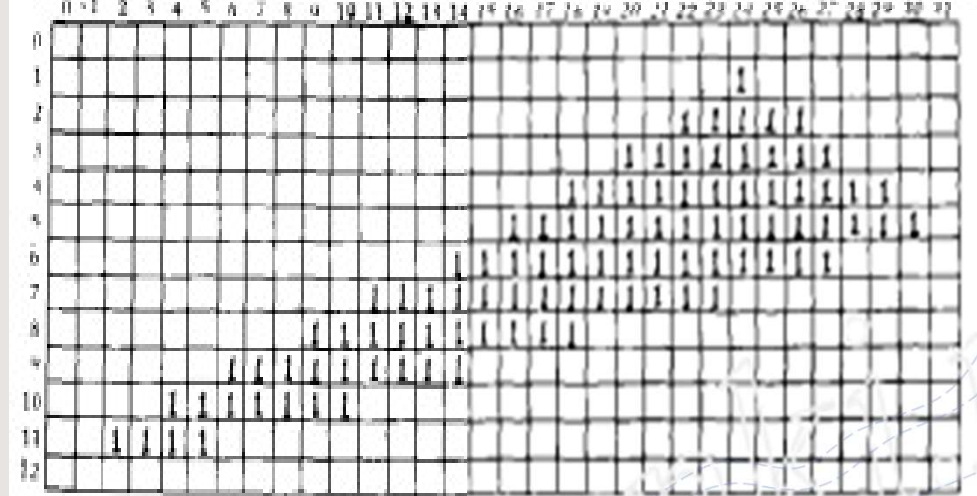
- + Fill the black sections
- + Scan each line of the marked dot matrix from left to right.
- + A depth counter checks whether B-marks and E-marks appear in pairs.



A 12x32 grid representing a marked dot matrix. The columns are indexed 0 to 31 and the rows 0 to 12. The matrix contains sparse pairs of '01' and '10' marks.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0																																
1																																
2																																
3																																
4																																
5																																
6																																
7																																
8																																
9																																
10																																
11																																
12																																

Marked Dot Matrix



A 12x32 grid representing the final dot matrix. The columns are indexed 0 to 31 and the rows 0 to 12. The matrix contains dense blocks of '1' marks, representing filled sections.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0																																
1																																
2																																
3																																
4																																
5																																
6																																
7																																
8																																
9																																
10																																
11																																
12																																

Final Dot Matrix

Basic Character Sizes

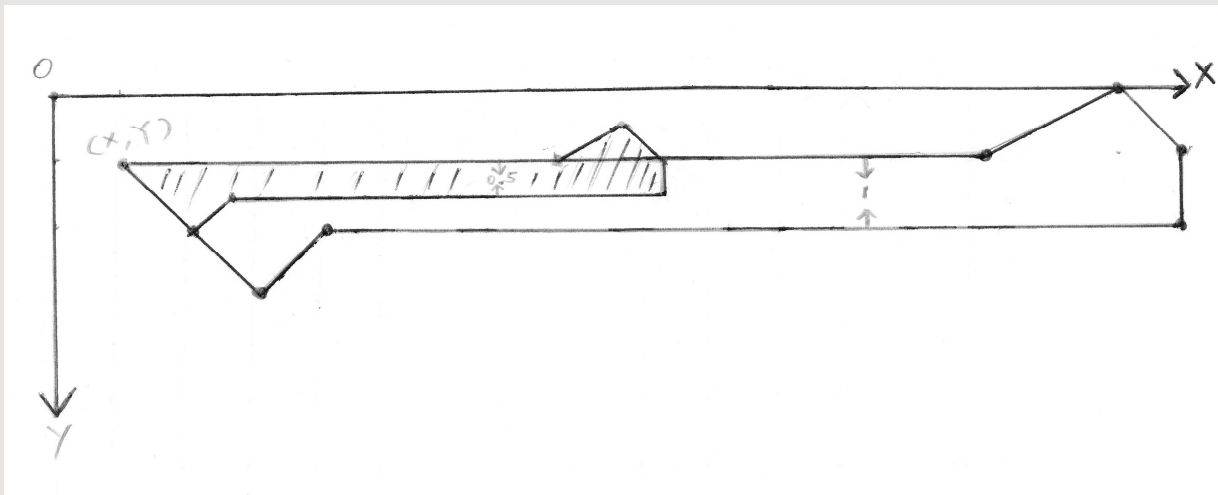
Character Size	Size of its grid
10.5pt	96 x 96
14pt	128x128
26pt	256x256

- + We do not need to create different compressed forms for the same character of different sizes.
- + Take 10.5pt, 14pt, and 26pt as our basic character sizes.
- + Use 14pt and 26pt as our basic character sizes for large characters used in titles or decorations.
- + If we scale up a character to a larger size, the grid will also expand to the corresponding larger grid.

Change the size of a character

* Scale factor = the desired size / a basic size

- (1) If a vector is related to the width of a regular stroke, we change the length of the vector so that the width of the stroke is equal to old width * scale factor.
- (2) For any vector that does not correspond to a regular stroke's width part, we multiply delta X and delta Y by the scale factor.



Summary

- + We talked about Chinese typesetting before the appearance of laser phototypesetting and introduced two typical laser phototypesetting systems.
- + British Monotype's Lasercomp uses the run-length coding.
- + HuaGuang system divides characters into compressible strokes.
- + In history, Professor Wang and his team improved the HG system and developed HG3, HG4, and Founder 91 based on this system.
- + Till 1993, the HG system and its derivations occupied most of the Chinese typesetting and printing market.
- + Till the end of the 20th century, all newspapers and publishing houses in China have adopted laser phototypesetting.

References

- [1] Wang Xuan; Lü Zhimin; Tang Yuhai; Xiang Yang;. A High Resolution Chinese Character Generator[J]., 1986, 1(2): 1-14.
- [2] 计算机-激光汉字编辑排版系统简介-《计算机学报》1981年02期-中国知网. (2021). Retrieved 9 July 2021, from <https://mall.cnki.net/magazine/article/JSJX198102000.htm>
- [3] 汉字激光照排机-《计算机学报》1981年02期-中国知网. (2021). Retrieved 9 July 2021, from <https://mall.cnki.net/magazine/Article/JSJX198102001.htm>
- [4] (2021). Retrieved 9 July 2021, from <https://pages.cpsc.ucalgary.ca/~gaines/reports/COMP/PenChina81/PenChina81.pdf>
- [5] Bakewell, B. (1984). The Benefits of Laser Phototypesetting and its Development on Lasercomp. The Journal Of Photographic Science, 32(4), 124-126. doi: 10.1080/00223638.1984.11738292
- [6] (2021). Retrieved 9 July 2021, from <https://patentimages.storage.googleapis.com/74/79/bf/03144d7257e790/EP0095536A1.pdf>
- [7] (2021). Retrieved 9 July 2021, from <https://apmub.files.wordpress.com/2014/01/monotype-and-phototypesetting-andrew-boag.pdf>

References

[8] (2021). Retrieved 9 July 2021, from

<https://web.archive.org/web/20151222115041/http://www.yzmuseum.com/whsb/view.asp?id=599>

[9] (2021). Retrieved 9 July 2021, from

<https://baike.baidu.com/item/%E9%9B%95%E7%89%88%E5%8D%B0%E5%88%B7%E6%9C%AF#5>

[10] (2021). Retrieved 9 July 2021, from

<https://zh.wikipedia.org/wiki/%E9%9B%95%E7%89%88%E5%8D%B0%E5%88%B7>

[11] (2021). Retrieved 9 July 2021, from

<https://web.archive.org/web/20151222115041/http://www.yzmuseum.com/whsb/view.asp?id=599>

[12] (2021). Retrieved 9 July 2021, from

<https://zh.wikipedia.org/wiki/%E4%B8%AD%E5%9B%BD%E6%9C%A8%E6%B4%BB%E5%AD%97%E5%8D%B0%E5%88%B7%E6%9C%AF>

[13] (2021). Retrieved 15 July 2021, from

<http://www.laibinyou.com/zixun/gsd/1279.html>

References

[14] (2021). Retrieved 15 July 2021, from

https://lh3.googleusercontent.com/proxy/-Ijjftxu33OOS38Kq3_cXNNgVbdoJHR8wcSjjnfOy-eQ0LakeBkKYsm2tPCaCcy7MCUdn7OEv6jyuqu0cUMo2evwVHSuGRiYnNGvtLA

[15] (2021). Retrieved 15 July 2021, from

[https://www.wikiwand.com/zh/%E7%8E%8B%E7%A5%AF_\(%E5%85%83%E6%9C%9D\)](https://www.wikiwand.com/zh/%E7%8E%8B%E7%A5%AF_(%E5%85%83%E6%9C%9D))

[16] Cheng chen. (1994). Hua guang bd pai ban yu yan shi yong da quan. Nan jing: Nan jing ta xue chu ban she.

[17] (2021). Retrieved 19 July 2021, from

http://www.kepuchina.cn/zt/2018/ggkf/201811/t20181119_822611.shtml