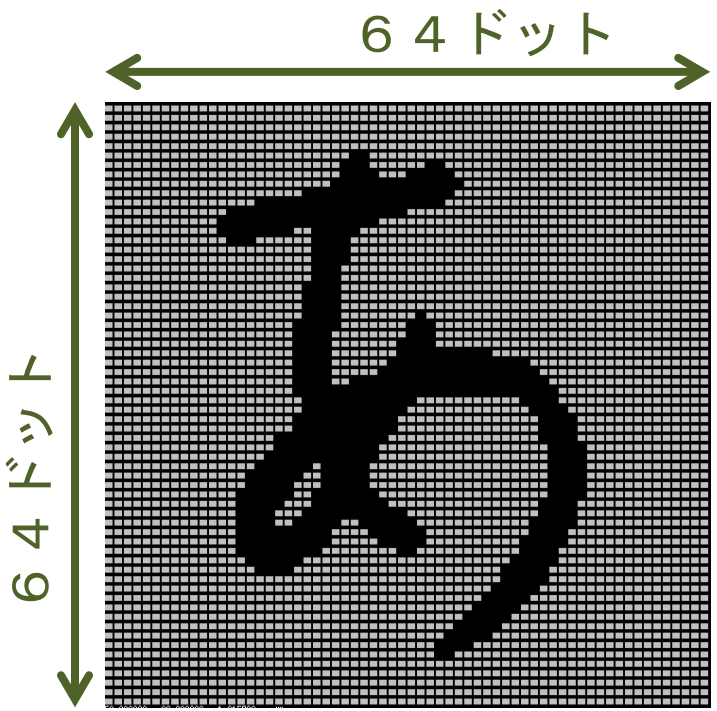
画像データの扱い方

知識工学Ⅰ

東京工業高等専門学校　情報工学科　鈴木雅人

文字画像の概要



- 一辺が64ドットの正方形画像
- モノクロ画像
- 背景を白，文字を黒で表現

一点の色情報は1ビットで表現可
（白を0，黒を1で表現する）

画像の点の数は64×64=4096ビット

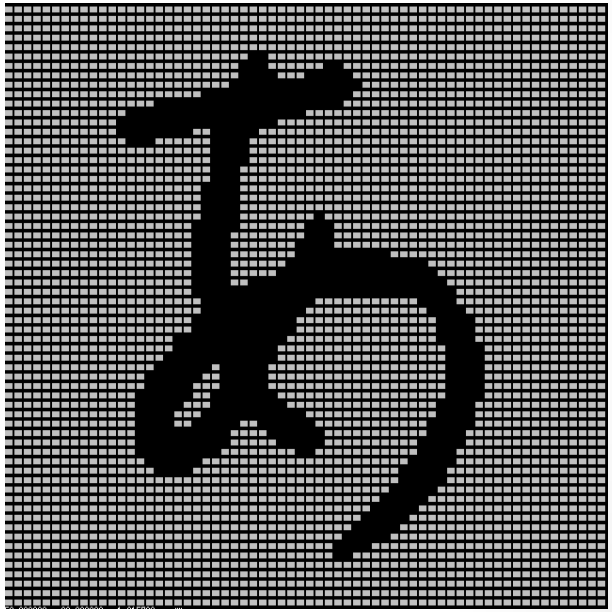
(画像サイズ)＝4096(bit)=4096/8(byte)=512(byte)

文字画像の品質

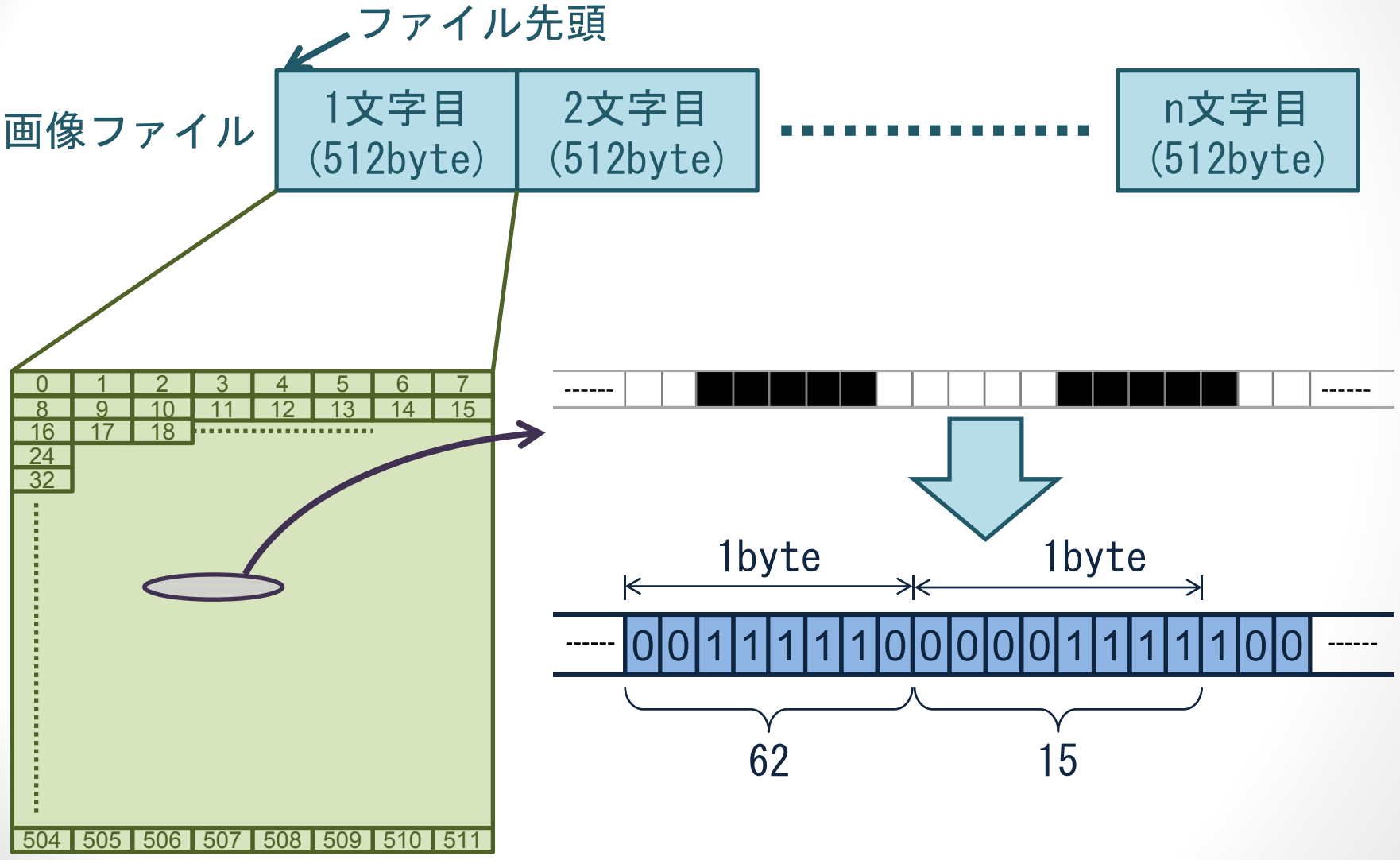
手書き文字のため多様な字形が想定されるが，「100%に近い多くの人が間違えずに読める」品質の文字画像を対象とする．

【特徴】

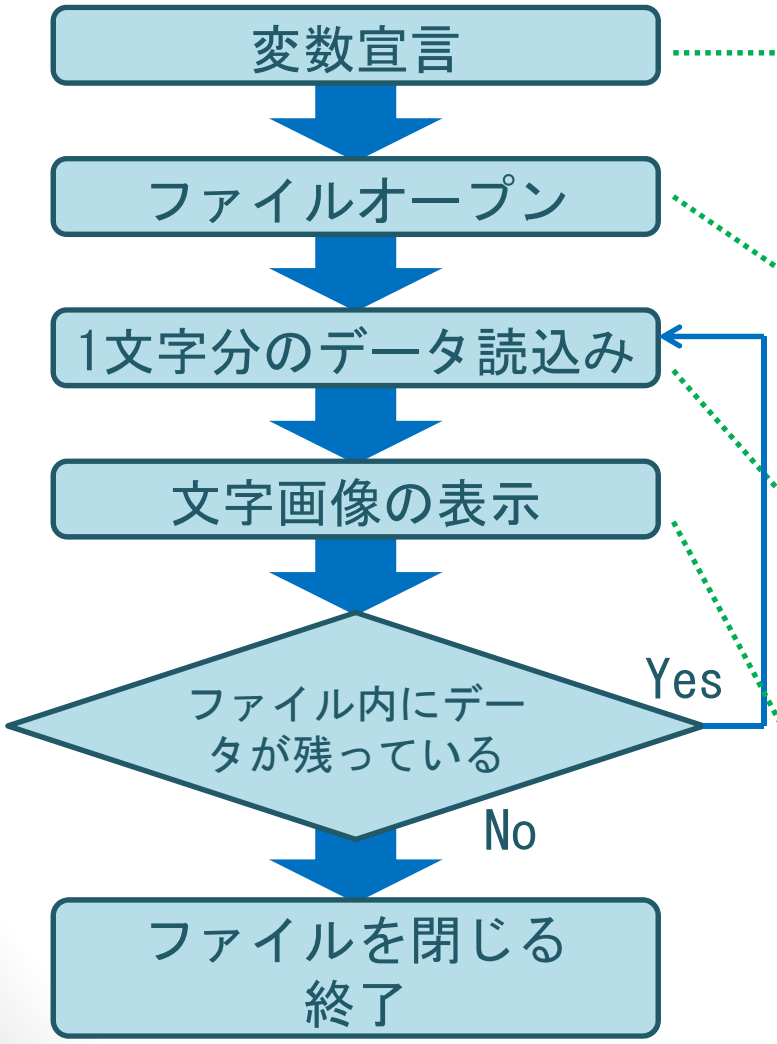
- 文字の形や大きさが一定でない．
- 文字が描かれている位置は必ずしも中央とは限らない．
- 文字は異なる字体で描かれることがある．
- 文字を構成する線の太さは，個々の文字によっても異る．また同じ文字でも部分的に線の太さが異なることもある．
- ノイズがのっている場合がある．



画像ファイルの形式


ファイル先頭
画像ファイル
1文字目
(512byte)
2文字目
(512byte)
n文字目
(512byte)
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
16 17 18
24
32
504 505 506 507 508 509 510 511
1byte
1byte
0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 0 0
62
15

画像表示プログラムの流れ



【Step1】
プログラムに必要な変数の宣言

【Step2】
fopen()関数の使い方

【Step3】
fread()関数の使い方

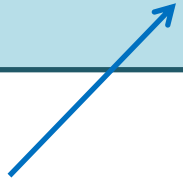
【Step4】
ビット演算を使った画像の表示
白画素を「o」黒画素を「*」で表現し，画面に表示する.

【step1】変数宣言

```
FILE   *fp ;          /* ファイル構造体のポインタ．画像
                         ファイルを開くために使う.
                      */

unsigned char  d[512] ;
                      /* 1文字分の画像データを入れる変数
                         符号あり型charの場合，右シフト演算
                         に不都合が生じるので符号なしchar型
                         を宣言している.
                         short int, long intでも処理は可能だ
                         がchar型で処理する方が小回りが利く
                      */
unsigned char  mask ; /* char型の8ビットデータから特定のビッ
                         トデータをとり出すために使用する.
                      */

int    i, x, y ;      /* ループで使う変数 */
```

【step2】fopen()関数

```
FILE    *fp ;

fp = fopen( “sample.img”, “rb” ) ;
```



r　…　読み込みのためのオープン
w　…　書き込みのためのオープン
b　…　バイナリファイルを扱う
　　（テキストでないファイル）

【step3】fread()関数

```
unsigned char d[512] ;
FILE  *fp ;

fread(  d,  512,  1,  fp ) ;
```

ファイルfpからデータを512バイトを1回読み込み変数d[]に代入する.

(返り値)

- データを読み込んだ回数.
- 上記の場合, 読み込みに成功すれば1, 失敗すれば0が返る.
  → 正しく読み込めたかどうかが判断できる.

【step4】画像の表示

まずd[0]に格納されている8ビット分のデータを表示する

```
mask = 0x80 ;  /* 0x80は16進数で80のこと．10進数で128．*/
               /* 2進数にすると「10000000」 */

if( d[0] & mask )          /* 8ビットのうち左端のビットを表示 */
   printf( “*” ) ;
else
   printf( “o” ) ;

mask = mask >> 1 ;         /* 「01000000」となるので左から2番目を
                              確かめることができる */
if( d[0] & mask )          /* 左から2番目のビットを表示 */
   printf( “*” ) ;
else
   printf( “o” ) ;         /* 以下同様に8回繰り返せば・・・ */
```

【step4】画像の表示

まずd[0]に格納されている8ビット分のデータを表示する

```
mask = 0x80 ;  /* 0x80は16進数で80のこと．10進数で128．*/
               /* 2進数にすると「10000000」 */

for( i = 0 ; i < 8 ; i++ ) {
    if( d[0] & mask ) {      /* maskはループを繰り返すたび1のビット
        printf( “*” ) ;          が右に移動する */
    } else {
        printf( “o” ) ;
    }
    mask = mask >> 1 ;
}
```

d[0]の0を1, 2, 3, …とすると，同じように8ビット分のデータが表示できる．

【step4】画像の表示

1行分(8byte=64bit)のデータを表示する.

```
mask = 0x80 ;  /* 0x80は16進数で80のこと. 10進数で128. */
               /* 2進数にすると「10000000」 */

for( x = 0 ; x < 8 ; x++ ) {
    mask = 0x80 ;
    for( i = 0 ; i < 8 ; i++ ) {
        if( d[x] & mask ) {
            printf( "*" ) ;
        } else {
            printf( "o" ) ;
        }
        mask = mask >> 1 ;
    }
}
```

/* maskはループを繰りすたびに
1のビットが右に移動する */

これを64回繰り返せば画像が表示できる.
2行目以降「x」の部分は何を書いたらいいか？

【step4】画像の表示

0行目	d[0] ～ d[7]
1行目	d[8] ～ d[15]
2行目	d[16] ～ d[23]
3行目	d[24] ～ d[31]
4行目	d[32] ～ d[39]
y行目	d[??] ～ d[??]
62行目	d[496]～d[503]
63行目	d[504]～d[511]

8の倍数？

d[8*y]～d[8*y+7]となりそう

ループの中のd[]内の式

y*8+x

【step4】画像の表示

```
for( y = 0 ; y < 64 ; y++ ) {
    for( x = 0 ; x < 8 ; x++ ) {
        mask = 0x80 ;
        for( i = 0 ; i < 8 ; i++ ) {
            if( d[y*8+x] & mask ) {
                printf( “*” ) ;
            } else {
                printf( “o” ) ;
            }
            mask = mask >> 1 ;
        }
    }
    printf( “¥n” ) ;
}
```