## 取扱説明書

```
PNM ファイルを読み取り、PBM、PGM、
image t* ReadPNM(char* input)
                            PPM イメージをサポートします。
                             以下の構造体を導入する必要がある:
                            typedef struct color t {
                              uint8 t r; //Red
                              uint8 t g; //Green
                              uint8 t b; //Blue
                              uint8 t a; //Alpha
                            } color t;
                            typedef union pixcel t {
                              color_t c; //RGBA
                              uint8 t g; //グレースケール
                              uint8 t i; //カラーインデックス
                            } pixcel_t;
                            typedef struct image t {
                              uint32 t width;
                                                 //幅
                              uint32 t height;
                                                 //高い
                              uint16_t color_type; //色の種類
                              uint16 t palette num; //パレット
                                                 //パレット
                              color_t *palette;
                             へのポインタ
                              pixcel t **map;
                                                 //画像デー
                            } image_t;
            WritePNM(image t*
                            PNM 画像データは画像ファイルとして
void
                            保存され、PBM、PGM、PPM 画像をサポー
input, char* output, int type)
                             トします。type は、type=1、2、3、4、
                            5、6 などの PNM ファイルのフォーマッ
                             トです。
                             以下の構造体を導入する必要がある:
                            typedef struct color t {
                              uint8_t r; //Red
                              uint8 t g; //Green
                              uint8_t b; //Blue
                              uint8 t a; //Alpha
                            } color t;
                            typedef union pixcel t {
                              color_t c; //RGBA
                              uint8 t g; //グレースケール
                              uint8 t i; //カラーインデックス
                            } pixcel t;
                             typedef struct image_t {
```

```
//幅
                              uint32 t width;
                                                 //高い
                              uint32 t height;
                              uint16_t color_type; //色の種類
                              uint16 t palette num; //パレット
                             数
                              color t *palette;
                                                  //パレット
                             へのポインタ
                              pixcel t **map;
                                                  //画像デー
                             } image_t;
image t* ReadBMP(char* input)
                             BMP画像を読み込みます。
                             以下の構造体を導入する必要がある:
                             typedef struct color t {
                              uint8_t r; //Red
                              uint8 t g; //Green
                              uint8 t b; //Blue
                              uint8 t a; //Alpha
                             } color_t;
                             typedef union pixcel t {
                              color t c; //RGBA
                              uint8 t g; //グレースケール
                              uint8 t i; //カラーインデックス
                             } pixcel t;
                             typedef struct image t {
                              uint32 t width;
                                                 //幅
                                                 //高い
                              uint32_t height;
                              uint16 t color type; //色の種類
                              uint16 t palette num; //パレット
                              color_t *palette;
                                                  //パレット
                             へのポインタ
                                                 //画像デー
                              pixcel t **map;
                             } image_t;
                             BMP 画像データは画像ファイルとして
void
            WriteBMP(image t*
input, char*
                             保存され、compress=1 の場合は RLE 圧
                  output, int
compress)
                             縮される。
                             以下の構造体を導入する必要がある:
                             typedef struct color t {
                              uint8_t r; //Red
                              uint8 t g; //Green
                              uint8 t b; //Blue
                              uint8_t a; //Alpha
                              color_t;
```

```
typedef union pixcel t {
                              color t c; //RGBA
                              uint8 t g; //グレースケール
                              uint8 t i; //カラーインデックス
                             } pixcel t;
                             typedef struct image t {
                              uint32 t width;
                                                 //幅
                              uint32 t height;
                                                 //高い
                              uint16_t color_type; //色の種類
                              uint16_t palette_num; //パレット
                              color_t *palette;
                                                  //パレット
                             へのポインタ
                              pixcel t **map;
                                                  //画像デー
                             タ
                             } image_t;
                             BMP 画像データは画像ファイルとして
void
            WriteBMP(image t*
                             保存される。
input, char* output)
                             以下の構造体を導入する必要がある:
                             typedef struct color t {
                              uint8 t r; //Red
                              uint8 t g; //Green
                              uint8 t b; //Blue
                              uint8_t a; //Alpha
                             } color t;
                             typedef union pixcel t {
                              color t c; //RGBA
                              uint8_t g; //グレースケール
                              uint8_t i; //カラーインデックス
                             } pixcel_t;
                             typedef struct image t {
                              uint32 t width;
                                                 //幅
                                                 //高い
                              uint32 t height;
                              uint16_t color_type; //色の種類
                              uint16 t palette num; //パレット
                             数
                              color t *palette;
                                                  //パレット
                             へのポインタ
                              pixcel t **map;
                                                  //画像デー
                             } image_t;
           WriteBMP1(image t*
                            BMP 画像データは画像ファイルとして
void
                             保存され、compress=1 の場合は RLE 圧
input, char*
                  output, int
                             縮される。
compress)
```

```
以下の構造体を導入する必要がある:
                              typedef struct color t {
                                uint8 t r; //Red
                                uint8 t g; //Green
                                uint8_t b; //Blue
                                uint8 t a; //Alpha
                              } color t;
                              typedef union pixcel t {
                                color t c; //RGBA
                                uint8 t g; //グレースケール
                                uint8 t i; //カラーインデックス
                              } pixcel t;
                              typedef struct image t {
                                                    //幅
                                uint32 t width;
                                uint32_t height;
                                                    //高い
                                uint16_t color_type; //色の種類
                                uint16_t palette_num; //パレット
                              数
                                                    //パレット
                                color t *palette;
                              へのポインタ
                                pixcel t **map;
                                                     //画像デー
                              タ
                              } image t;
                              多焦点画像の融合、8 ビット BMP 画像
void
             ImageFusion(char*
input1, char*
                              をサポートする。block height=8,
                  input2, char*
output, int
              block_height, int
                              block width=8, threshold=1.75.
block width, double threshold)
                              画像融合。リファレンス: a=3, b1=4,
void
             ImageFusion(char*
input1, char*
                  input2, char*
                              DX1=-68, DY1=-99, EPS=1, input1="画
MaskImage, char*
                              像融合 1. jpg", input2="画像融合
                   output, int
          dy[], int
                              2. jpg", MaskImage="マスク.png",
dx[], int
                     a, double
                              output="output.jpg".
b1, int DX1, int DY1, double EPS)
                              int dx[] = \{0, 0, -1, 1\};
                              int dy[] = \{-1, 1, 0, 0\};
void
             ImageFusion(char*
                              画像融合、PNG 画像をサポートする。リ
input1, char*
                  input2, char*
                              ファレンス: input1="
                                                     画像融合
                              1.png", input2="
                                                   画像融合
inputUniqe1, char*
inputUniqe2, char* output)
                              2. png", inputUniqe1="
                                                      画像融合
                              1 unique.txt", inputUniqe2=" 画
                              像融合 2_unique.txt"。
                              画像融合、PNG 画像をサポートする。リ
void
     Uniqe(char*
                  input, char*
inputUniqe, char* output, double
                              ファレンス: input="
                                                     画像融合
R, double G, double B)
                              1. png", inputUniqe="
                                                      画像融合
                              1 unique.txt" . R=255, G=0, B=0.
```

void Screenshot1(HWND hWnd, LPCWSTR OutputImage)
void Screenshot2(HWND hWnd, LPCWSTR OutputImage)
void Screenshot2(HWND hWnd, LPCWSTR OutputImage)
void Screenshot2(HWND hWnd, LPCWSTR Output Image) ショットするウィンドウハンドルです。たとえば、次のようになります: GetDesktopWindow(): Output Image はスクリーンショット名です。 void Screenshot3(HWND hWnd, スクリーンショット名です。  void Screenshot3(HWND hWnd, スクリーンショット名です。  void Screenshot3(HWND hWnd, スクリーンショット名です。  void Screenshot3(HWND hWnd, スクリーンショット名です。  uint8_t* AESencrypt (uint8_t* input, uint8_t* key, int size)
Void   Screenshot2(HWND hWnd, LPCWSTR Output Image)
hWnd, LPCWSTR OutputImage)
void Screenshot3(HWND hWnd, LPCWSTR OutputImage)  viint8_t* AESencrypt(uint8_t* input, uint8_t* key, int size)  wint DES_Encrypt(char *PlainFile, char *Key, char *CipherFile, char *Key, char *PlainFile, char *Key, char *CipherFile the Get Compute the Get Capen the Cap
void Screenshot3(HWND hWnd, スクリーンショット名です。  void Screenshot3(HWND hWnd, スクリーン関数。hWnd は、スクリーンショットするウィンドウハンドルです。たとえば、次のようになります: GetDesktopWindow(); OutputImage はスクリーンショット名です。  uint8_t* AESencrypt(uint8_t* input, uint8_t* key, int size)  uint8_t* AESdecrypt(uint8_t* input, uint8_t* key, int size)  wint8_t* AESdecrypt(uint8_t* input, uint8_t* key, int size)  void DES_Encrypt(char **PlainFile, char *Key, char **CipherFile は暗号化されたアイル名です。  void DES_Decrypt(char **PlainFile, char *Key, char **CipherFile は暗号化されたファイルをサポートします。PlainFile は暗号化されたファイル名です。  void DES_Decrypt(char **PlainFile, char **Key, char **Tートします。CipherFile は暗号化されたファイル名です。  void DES_Decrypt(char **PlainFile, char **Key, char **Tートします。CipherFile は暗号化されたファイル名です。  int Equal(char* input1, char* input2, double c)  wid Equal(char* input1, char* timput2, double c)  wid Ethexhow (); OutputImage は スクリーン とも、大きなは、次のようになります: GetDesktopWindow(); OutputImage は スクリーンを引きます。  wint8_t* AESencrypt(uint8_t*
void Screenshot3(HWND hWnd, LPCWSTR OutputImage) ショットするウィンドウハンドルです。たとえば、次のようになります: GetDesktopWindow(); OutputImage はスクリーンショット名です。  uint8_t* AESencrypt(uint8_t* key, int size) は鍵、size は input のサイズです。暗号化結果データを返します。  uint8_t* AESdecrypt(uint8_t* key, int size) は鍵、size は input のサイズです。暗号化結果データを返します。  void DES_Encrypt(char **Rey, char **PlainFile, char **Key, char **CipherFile, char **Key, char **CipherFile, char **Key, char **PlainFile)
Void   Screenshot3 (HWND   hWnd, LPCWSTR Output Image)   ショットするウィンドウハンドルです。たとえば、次のようになります: GetDesktopWindow(); Output Image はスクリーンショット名です。   Wint8_t*   AESencrypt (wint8_t*   key, int size)   は鍵、size は input のサイズです。暗号化結果データを返します。   Wint8_t*   AESdecrypt (wint8_t*   key, int size)   という   Wind   Win
LPCWSTR OutputImage)  ショットするウィンドウハンドルで す。たとえば、次のようになります: GetDesktopWindow(): OutputImage は スクリーンショット名です。  uint8_t* AESencrypt(uint8_t* input, uint8_t* key, int size)  uint8_t* AESdecrypt(uint8_t* input, uint8_t* key, int size)  DES 復号関数、input は暗号化されたデータを返します。  Wind DES_Encrypt(char **PlainFile, char **Key, char **CipherFile)  **CipherFile, char **Key, char **CipherFile, char **Key, char **PlainFile, char **Key, char **PlainFile, char **Key, char **PlainFile, char **Key, char **CipherFile, char **Key, char **PlainFile は暗号化されたファイル名です。  **Void DES_Decrypt(char **CipherFile, char **Key, char **CipherFile は暗号化されたファイル名です。  **Void DES_Decrypt(char **CipherFile は暗号化されたファイル名です。  **Void DES_Decrypt(char **CipherFile は暗号化されたファイル名です。  **Void DES_Decrypt(char **CipherFile は暗号化されたファイルのファイル名、Key はキー文字、PlainFile は復号後のファイル名です。  int Equal(char* input1, char* input2, double c)  int Equal(char* input1, char* input2, double c)
### description of the imput, uint8_t* AESencrypt (uint8_t* aesencrypt
### GetDesktopWindow(): OutputImage は スクリーンショット名です。  ###################################
uint8_t* AESencrypt(uint8_t* dbs. aES 暗号化関数、input は元データ、key は鍵、size は input のサイズです。暗号化結果データを返します。  uint8_t* AESdecrypt(uint8_t* dbs. aES 復号関数、input は暗号化されたデータ、key は鍵、size は input のサイズです。復号結果データを返します。  void DES_Encrypt(char **Rey, char **Key, char **CipherFile)  void DES_Decrypt(char **CipherFile, char **Key, char **CipherFile, char **Key, char **CipherFile, char **Key, char **PlainFile)  void DES_Decrypt(char **CipherFile は暗号化されたファイル名です。  void DES_Decrypt(char **CipherFile は暗号化されたファイル名です。  void DES_Decrypt(char **CipherFile は暗号化されたファイル名です。  void DES_Decrypt(char **CipherFile は暗号化されたファイル名です。  int Equal(char* input1, char* input1, char* は画像の勾配振幅類似性偏差値が c に 等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は 参照の閾値である。24 ピット BMP 画像 をサポートしています。
wint8_t* AESencrypt(uint8_t* input, uint8_t* key, int size) は鍵、size は input のサイズです。暗号化結果データを返します。 wint8_t* AESdecrypt(uint8_t* input, uint8_t* key, int size)
input, uint8_t* key, int size) は鍵、size は input のサイズです。暗号化結果データを返します。  uint8_t* AESdecrypt (uint8_t*
B代結果データを返します。 uint8_t* AESdecrypt (uint8_t* AES 復号関数、input は暗号化されたデinput, uint8_t* key, int size)
wint8_t* AESdecrypt (wint8_t* input, wint8_t* key, int size) input, wint8_t* key, int size)  void DES_Encrypt (char *PlainFile, char *Key, char *PlainFile)  void DES_Decrypt (char *CipherFile)  void DES_Decrypt (char *CipherFile, char *Key, char *PlainFile, char *Key, char *PlainFile, char *Key, char *PlainFile)  void DES_Decrypt (char *CipherFile, char *Key, char *PlainFile)  void DES_Decrypt (char *CipherFile は暗号化されたファイル名です。  void DES_Decrypt (char *CipherFile は暗号化されたファイル名です。  void DES_Decrypt (char *CipherFile は暗号化されたファイル名です。  int Equal (char* input1, char* input1, char* 比画像の勾配振幅類似性偏差値がcに等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は参照の閾値である。24 ビット BMP 画像をサポートしています。
input, uint8_t* key, int size)
void DES_Encrypt(char という) DES Encrypt(char を表します。 DES 暗号化関数で、複数のファイルを *PlainFile, char *Key, char *ボートします。PlainFile は元のファイル名、Key はキー文字、 CipherFile は暗号化されたファイル名です。  void DES_Decrypt(char *CipherFile, char *Key, char *ボートします。CipherFile は暗号化されたファイルをサポートします。CipherFile は暗号化されたファイル名です。  *PlainFile) が、たファイルのファイル名、Key はキー文字、PlainFile は復号後のファイル名です。  int Equal(char* input1, char* 比画像の勾配振幅類似性偏差値が c に 等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は 参照の閾値である。24 ビット BMP 画像をサポートしています。
void DES_Encrypt (char *PlainFile, char *Key, char *CipherFile)
*PlainFile, char *Key, char *CipherFile) ** *CipherFile) ** *CipherFile は暗号化されたファイル名です。  void DES_Decrypt(char *CipherFile, char *Key, char *PlainFile) ** *PlainFile) ** *PlainFile) ** *The Late CipherFile は暗号化されたファイルをサポートします。CipherFile は暗号化されたファイルのファイル名、Key はキー文字、PlainFile は復号後のファイル名です。  int Equal(char* input1, char* は画像の勾配振幅類似性偏差値がではいます。  input2, double c) ** *Equal(char* input1, char* は画像の勾配振幅類似性偏差値がではいます。  などは比較対象の2つの画像である。とは、参照の閾値である。24 ビット BMP 画像をサポートしています。
*CipherFile
CipherFile は暗号化されたファイル名です。  void DES_Decrypt(char *CipherFile, char *Key, char *PlainFile) ポートします。CipherFile は暗号化されたファイルのファイル名、Key はキー文字、PlainFile は復号後のファイル名です。 int Equal(char* input1, char* input2, double c)
Void   DES_Decrypt(char   Void   DES_Decrypt(char   Void   DES_Decrypt(char   Void   Notation   Void   Notation   Void   Notation   Void   Notation   Void   Vo
void DES_Decrypt(char *CipherFile, char *Key, char *ポートします。CipherFile は暗号化さ *PlainFile) ** ** ** ** ** ** ** ** ** ** ** ** **
*CipherFile, char *Key, char *Rey, char *PlainFile) ポートします。CipherFile は暗号化されたファイルのファイル名、Key はキー文字、PlainFile は復号後のファイル名です。  int Equal(char* input1, char* 比画像の勾配振幅類似性偏差値が c に等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は参照の閾値である。24 ビット BMP 画像をサポートしています。
*PlainFile) れたファイルのファイル名、Key はキー文字、PlainFile は復号後のファイル名です。 int Equal(char* input1, char* 比画像の勾配振幅類似性偏差値が c に等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は参照の閾値である。24 ビット BMP 画像をサポートしています。
一文字、PlainFile は復号後のファイル名です。   int Equal(char* input1, char* 比画像の勾配振幅類似性偏差値が c に 等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は 参照の閾値である。24 ビット BMP 画像 をサポートしています。
n 名です。 int Equal(char* input1, char* 比画像の勾配振幅類似性偏差値が c に input2, double c) 等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は 参照の閾値である。24 ビット BMP 画像 をサポートしています。
int Equal(char* input1, char* 比画像の勾配振幅類似性偏差値が c に 等しければ通過する。input 1 と input 2 は比較対象の 2 つの画像である。c は 参照の閾値である。24 ビット BMP 画像 をサポートしています。
input2, double c)等しければ通過する。input 1 と input2 は比較対象の2つの画像である。c は参照の閾値である。24 ビット BMP 画像をサポートしています。
2 は比較対象の 2 つの画像である。c は 参照の閾値である。24 ビット BMP 画像 をサポートしています。
参照の閾値である。24 ビット BMP 画像 をサポートしています。
をサポートしています。
int GreaterThan(char* 比画像の勾配振幅類似性偏差値が c よ
input1, char* input2, double c) り大きければ通過する。input 1 と
input 2は比較対象の2つの画像であ
る。c は参照の閾値である。24 ビット
BMP 画像をサポートしています。
int LessThan(char* input1, char* 比画像の勾配振幅類似性偏差値が c 未
input2, double c) 満であれば通過する。input 1 と input
2は比較対象の2つの画像である。cは

	参照の閾値である。24 ビット BMP 画像
	をサポートしています。
double GMSD(char* input1, char*	2 枚の画像の勾配振幅類似性偏差値を
input2)	求め、結果を返す。input 1 と input 2
	は比較対象の2つの画像である。24ビ
	ット BMP 画像をサポートしています。
void FileWrite(char* BMP, char*	画像が暗黙的に書かれたファイルが書
TXT)	き込まれ、テキストファイルが画像に
	書き込まれる。32 ビット BMP 画像をサ
	ポートしています。BMP は書き込む画
	像ファイル名であり、TXT は画像を書
	き込むテキストファイル名である。
void FileWriteOut(char*	画像を隠して書いたファイルを書き出
BMP, char* TXT)	し、テキストファイルを画像から取り
	出します。32 ビット BMP 画像をサポー
	トしています。BMP は書き出す画像フ
	ァイル名であり、TXT は画像を書き出
	すと情報が保存されるテキストファイ
	ル名である。
void Watershed2(char*	画像分割の分水嶺アルゴリズム。
input, char*	inputMarqueurs は入力画像のマーキ
inputMarqueurs, char* output, int	ング画像である。R=230, G=0, B=0, r=1。
r,unsigned char R,unsigned char	24ビットBMP画像をサポートしていま
G, unsigned char B)	す。
void EcrireImage1(char*	画像分割。rayon=5。24 ビット BMP 画
input, char* output, uint32_t	像をサポートしています。
rayon)	
void EcrireImage2(char*	
input, char*	像をサポートしています。
inputMarqueurs,char*	
output, uint32_t rayon)	
void EcrireLPECouleur1(char*	画像分割。rayon=5。24 ビット BMP 画
input, char*	像をサポートしています。
inputMarqueurs,char*	
output, uint32_t rayon)	
void Watershed1(char*	画像分割の分水嶺アルゴリズム。
input, char*	inputMarqueurs は入力画像のマーキ
inputMarqueurs,char*	ング画像である。rayon=5。24 ビット
output, uint32_t rayon)	BMP 画像をサポートしています。
void EcrireImage3(char*	画像分割。rayon=1。24 ビット BMP 画
input, char*	像をサポートしています。
inputMarqueurs,char*	
output, uint16_t rayon)	
void	画像分割。rayon=1。24 ビット BMP 画

EcrireImageCouleursAleatoires(c	像をサポートしています。
har* input, char*	
inputMarqueurs, char*	
output, uint8_t r, uint8_t	
g,uint8_t b,uint16_t rayon)	
void Watershed(char*	画像分割の分水嶺アルゴリズム。
input, char*	inputMarqueurs は入力画像のマーキ
inputMarqueurs, char*	ング画像である。a は一般的に 255、
output, uint8 t r, uint8 t	rayon=1 である。24 ビット BMP 画像を
g, uint8_t b, uint8_t a, uint16_t	
rayon)	γ ν γ σ ε ν ω γ σ
double	文字マッチング、BMP 画像をサポート
CharacterRecognition(char*	し、戻り値はターゲット画像がマッチ
Target Image, char*	ングしたテンプレートファイルのシー
TemplateFileGroup[])	ケンス番号であり、戻り値が2であれ
Tembrace. Treoroub[])	ば画像とシーケンス番号が 2 (シーケー
	ンス番号がゼロから始まる)のテンプ
	レートのマッチングを説明する。
	リ ファレンス:
	<pre>TemplateFileGroup[]={</pre>
	"4. txt", "5. txt", "6. txt",
	"7. txt", "8. txt", "9. txt" };
double	文字マッチング、BMP 画像をサポート
CharacterRecognition1(char*	し、戻り値はターゲット画像がマッチ
TargetImage, char*	ングしたテンプレートファイルのシー
TemplateFileGroup[])	ケンス番号であり、戻り値が2であれ
	ば画像とシーケンス番号が 2 (シーケ
	ンス番号がゼロから始まる)のテンプ
	レートのマッチングを説明する。
	リ ファ レ ン ス :
	TemplateFileGroup[]={ "0.txt",
	"1. txt", "2. txt", "3. txt",
	"4. txt", "5. txt", "6. txt",
	"7. txt", "8. txt", "9. txt" };
void	文字分割。BMP 画像をサポートしてい
CharacterSegmentation(char*	ます。
input, string OutputFolder, int	OutputFolder は結果出力のフォルダ
YHistogramValleyMaxPixelNumber,	であり、「output」のように、結果を出
int	力するファイル名の構成方法は、左上
XHistogramValleyMaxPixelNumber,	の X 座標-左上の Y 座標-右下の X 座標
double	-右下の Y 座標、
SubImgBlackPixelPercentage, int	YHistogramValleyMaxPixelNumberはY
SingleNumberImgBoundary, int	方向ヒストグラムを求めるので、谷の

最少の黒い画素の個数, YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber は X 方向ヒストグラムを求めるので、谷の 最少の黒い画素の個数, XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage は、サブ マップ内の黒のピクセルが一定パーセ ントを超えている場合にのみ数値にな ま す SubImgBlackPixelPercentage=0.001, SingleNumberImgBoundary は、1 枚のデ ジタル画像エッジの塗り幅です, SingleNumberImgBoundary=5, Infinite は無限大とみなす,Infinite=249480, NumberImageBlackPixelPercentage は、1 枚のデジタル画像の黒画素数が すべてのデジタル画像を上回る, NumberImageBlackPixelPercentage=0. 35。

void

CharacterSegmentation(char\* input, char\* output, int BoundaryRemoveGap, int BinaryGap, int YHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int Infinite, int XHistogramValleyMaxPixelNumber, double NumberImageBlackPixelPercentage , int SingleNumberImgBoundary)

文字分割。BMP 画像をサポートしてい ます。

BinaryGap は画像二値化グローバル閾 値 で あ る , BinaryGap=135 , BoundaryRemoveGap はエッジがすべて 白に設定された距離です, BoundaryRemoveGap=7,インフィニット は無限大とみなす, Infinite=249480, SingleNumberImgBoundary は、1 枚のデ ジタル画像エッジの塗り幅です, SingleNumberImgBoundary=5 YHistogramValleyMaxPixelNumber は Y 方向ヒストグラムを求めるので、谷の 最少の黒い画素の個数, YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber は X 方向ヒストグラムを求めるので、谷の 最少の黒い画素の個数, XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage は、サブ マップ内の黒のピクセルが一定パーセ ントを超えている場合にのみ数値にな SubImgBlackPixelPercentage=0.001,

	NumberImageBlackPixelPercentage
	は、1枚のデジタル画像の黒画素数が
	すべてのデジタル画像を上回る,
	NumberImageBlackPixelPercentage=0.
	35.
	リファレンス: output="output"。
void CodeEncoding(std::string	
input, char* output, int	る文字列であり、output は生成される
width, int height, int margin,	2次元コード画像ファイル名である。
int eccLevel, int stride bytes,	margin: バーコード周辺のマージン
int comp, int a)	ecc: 誤り訂正レベル, [0-8]
The comp, the a)	a=1: AZTEC
	a=1: AZTEC a=2: CODABAR
	a=3: CODE_39
	a=4: CODE_93
	a=5: CODE_128
	a=6: DATA_MATRIX
	a=7: EAN_8
	a=8: EAN_13
	a=9: ITF
	a=10: MAXICODE
	a=11: PDF_417
	a=12: QR_CODE
	a=13: RSS_14
	a=14: RSS_EXPANDED
	a=15: UPC_A
	a=16: UPC_E
	a=17: UPC_EAN_EXTENSION
	リファレンス: margin=10, eccLevel=-
	1, stride_bytes=0, comp=1.
std::string CodeDecoding(char*	2次元コード復号。input は入力された
<pre>input, int req_comp, int a)</pre>	2 次元コード画像ファイル名であり、
	復号結果を返す。
	a=1: Lum
	a=2: RGB
	a=3: BGR
	a=4: RGBX
	a=5: XRGB
	a=6: BGRX
	a=7: XBGR
	リファレンス: req_comp=4, a=4。
	, ,