取扱説明書

void ImageFusion(char* input1,char* input2,char* MaskImage,char* output,int dx[],int dy[],int a,double b1,int DX1,int DY1,double EPS)	画像融合。リファレンス：a=3，b1=4，DX1=-68，DY1=-99，EPS=1，input1="画像融合 1.jpg"，input2="画像融合 2.jpg"，MaskImage="マスク.png"，output="output.jpg"。 int dx[] = {0,0,-1,1}; int dy[] = {-1,1,0,0};
void DES_Encrypt(char *PlainFile, char *Key,char *CipherFile)	DES 暗号化関数で、複数のファイルをサポートします。PlainFile は元のファイルのファイル名、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 *PlainFile)	DES 復号関数は、複数の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。
void FileWrite(char* BMP,char* TXT)	画像が暗黙的に書かれたファイルが書き込まれ、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* BMP,char* TXT)	画像を隠して書いたファイルを書き出し、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* input,char* inputMarqueurs,char* output,int r,unsigned char R,unsigned char G,unsigned char B)	画像分割の分水嶺アルゴリズム。inputMarqueurs は入力画像のマーキング画像である。R=230，G=0，B=0，r=1。24 ビット BMP 画像をサポートしています。
void EcrireImage1(char* input,char* output,uint32_t rayon)	画像分割。rayon=5。PNG 画像をサポートしています。
void EcrireImage2(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	画像分割。rayon=5。PNG 画像をサポートしています。
void EcrireLPECouleur1(char*	画像分割。rayon=5。PNG 画像をサポー

input,char* inputMarqueurs,char* output,uint32_t rayon)	トしています。
void Watershed1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	画像分割の分水嶺アルゴリズム。inputMarqueurs は入力画像のマーキング画像である。rayon=5。PNG 画像をサポートしています。
void EcrireImage3(char* input,char* inputMarqueurs,char* output,uint16_t rayon)	画像分割。rayon=1。PNG イメージに対応しています。
void EcrireImageCouleursAleatoires(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint16_t rayon)	画像分割。rayon=1。PNG イメージに対応しています。
void Watershed(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint8_t a,uint16_t rayon)	画像分割の分水嶺アルゴリズム。inputMarqueurs は入力画像のマーキング画像である。a は一般的に 255、rayon=1 である。PNG イメージに対応しています。
double CharacterRecognition(char* TargetImage,char* TemplateFileGroup[])	文字マッチング、BMP 画像をサポートし、戻り値はターゲット画像がマッチングしたテンプレートファイルのシーケンス番号であり、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" };
double CharacterRecognition1(char* TargetImage,char* TemplateFileGroup[])	文字マッチング、BMP 画像をサポートし、戻り値はターゲット画像がマッチングしたテンプレートファイルのシーケンス番号であり、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 CharacterSegmentation(char* input, string OutputFolder, int YHistogramValleyMaxPixelNumber, int XHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int SingleNumberImgBoundary, int Infinite, double NumberImageBlackPixelPercentage)	文字分割。BMP 画像をサポートしています。 OutputFolder は結果出力のフォルダであり、「output」のように、結果を出力するファイル名の構成方法は、左上の X 座標-左上の Y 座標-右下の X 座標-右下の Y 座標、 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ります，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	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

, int SingleNumberImgBoundary)	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ります，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 width,int height, int margin, int eccLevel, int stride_bytes, int comp,int a)	2 次元コード符号化。input は符号化する文字列であり、output は生成される 2 次元コード画像ファイル名である。 margin：バーコード周辺のマージン ecc：誤り訂正レベル，[0-8] 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* input,int req_comp,int a)	2 次元コード復号。input は入力された 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。