**使用说明书**

|  |  |
| --- | --- |
| image\_t\* ReadPNM(char\* input) | 读取PNM文件，支持PBM、PGM和PPM图像。  需引入以下结构体：  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; //灰度  uint8\_t i; //颜色索引  } pixcel\_t;  typedef struct image\_t {  uint32\_t width; //宽  uint32\_t height; //高  uint16\_t color\_type; //颜色种类  uint16\_t palette\_num; //调色板数量  color\_t \*palette; //指向调色板的指针  pixcel\_t \*\*map; //图像数据  } image\_t; |
| void WritePNM(image\_t\* input,char\* output,int type) | PNM图像数据保存为图像文件，支持PBM、PGM和PPM图像，type是PNM文件的格式，如type=1、2、3、4、5、6。  需引入以下结构体：  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; //灰度  uint8\_t i; //颜色索引  } pixcel\_t;  typedef struct image\_t {  uint32\_t width; //宽  uint32\_t height; //高  uint16\_t color\_type; //颜色种类  uint16\_t palette\_num; //调色板数量  color\_t \*palette; //指向调色板的指针  pixcel\_t \*\*map; //图像数据  } image\_t; |
| image\_t\* ReadBMP(char\* input) | 读取BMP图像。  需引入以下结构体：  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; //灰度  uint8\_t i; //颜色索引  } pixcel\_t;  typedef struct image\_t {  uint32\_t width; //宽  uint32\_t height; //高  uint16\_t color\_type; //颜色种类  uint16\_t palette\_num; //调色板数量  color\_t \*palette; //指向调色板的指针  pixcel\_t \*\*map; //图像数据  } image\_t; |
| void WriteBMP(image\_t\* input,char\* output,int compress) | BMP图像数据保存为图像文件，compress=1时进行RLE压缩。  需引入以下结构体：  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; //灰度  uint8\_t i; //颜色索引  } pixcel\_t;  typedef struct image\_t {  uint32\_t width; //宽  uint32\_t height; //高  uint16\_t color\_type; //颜色种类  uint16\_t palette\_num; //调色板数量  color\_t \*palette; //指向调色板的指针  pixcel\_t \*\*map; //图像数据  } image\_t; |
| void WriteBMP(image\_t\* input,char\* output) | BMP图像数据保存为图像文件。  需引入以下结构体：  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; //灰度  uint8\_t i; //颜色索引  } pixcel\_t;  typedef struct image\_t {  uint32\_t width; //宽  uint32\_t height; //高  uint16\_t color\_type; //颜色种类  uint16\_t palette\_num; //调色板数量  color\_t \*palette; //指向调色板的指针  pixcel\_t \*\*map; //图像数据  } image\_t; |
| void WriteBMP1(image\_t\* input,char\* output,int compress) | BMP图像数据保存为图像文件，compress=1时进行RLE压缩。  需引入以下结构体：  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; //灰度  uint8\_t i; //颜色索引  } pixcel\_t;  typedef struct image\_t {  uint32\_t width; //宽  uint32\_t height; //高  uint16\_t color\_type; //颜色种类  uint16\_t palette\_num; //调色板数量  color\_t \*palette; //指向调色板的指针  pixcel\_t \*\*map; //图像数据  } image\_t; |
| void ImageFusion(char\* input1,char\* input2,char\* output,int block\_height,int block\_width,double threshold) | 多聚焦图像的融合，支持8位BMP图像。block\_height=8，block\_width=8，threshold=1.75。 |
| void ImageFusion(char\* input1,char\* input2,char\* MaskImage,char\* output,int dx[],int dy[],int a,double b1,int DX1,int DY1,double EPS) | 图像融合。参考：a=3，b1=4，DX1=-68，DY1=-99，EPS=1，input1="图像融合1.jpg"，input2="图像融合2.jpg"，MaskImage="掩膜.png"，output="output.jpg"。  int dx[] = {0,0,-1,1};  int dy[] = {-1,1,0,0}; |
| void ImageFusion(char\* input1,char\* input2,char\* inputUniqe1,char\* inputUniqe2,char\* output) | 图像融合，支持PNG图像。参考：input1=”图像融合1.png”，input2=”图像融合2.png”， inputUniqe1=” 图像融合1\_unique.txt”， inputUniqe2=” 图像融合2\_unique.txt”。 |
| void Uniqe(char\* input,char\* inputUniqe,char\* output,double R,double G,double B) | 图像融合，支持PNG图像。参考：input=”图像融合1.png”， inputUniqe=” 图像融合1\_unique.txt”。R=255，G=0，B=0。 |
| void Screenshot1(HWND hWnd, LPCWSTR OutputImage) | 截屏函数。hWnd是要截屏的窗口句柄，如：GetDesktopWindow()；OutputImage 截图名称。 |
| void Screenshot2(HWND hWnd,LPCWSTR OutputImage) | 截屏函数。hWnd是要截屏的窗口句柄，如：GetDesktopWindow()；OutputImage 截图名称。 |
| void Screenshot3(HWND hWnd, LPCWSTR OutputImage) | 截屏函数。hWnd是要截屏的窗口句柄，如：GetDesktopWindow()；OutputImage 截图名称。 |
| uint8\_t\* AESencrypt(uint8\_t\* input,uint8\_t\* key,int size) | AES加密函数，input是原数据，key是密钥，size是input的大小。返回加密结果数据。 |
| uint8\_t\* AESdecrypt(uint8\_t\* input,uint8\_t\* key,int size) | AES解密函数，input是已加密数据，key是密钥，size是input的大小。返回解密结果数据。 |
| void DES\_Encrypt(char \*PlainFile, char \*Key,char \*CipherFile) | DES加密函数，支持多种文件。PlainFile是原文件的文件名，Key是密钥字符，CipherFile是加密后的文件名。 |
| void DES\_Decrypt(char \*CipherFile, char \*Key,char \*PlainFile) | DES解密函数，支持多种文件。CipherFile是已加密文件的文件名，Key是密钥字符，PlainFile是解密后的文件名。 |
| int Equal(char\* input1,char\* input2,double c) | 若比对图像的梯度幅相似性偏差值等于c则通过。input1和input2是要比对的两个图像。c是参考的阈值。支持24位BMP图像。 |
| int GreaterThan(char\* input1,char\* input2,double c) | 若比对图像的梯度幅相似性偏差值大于c则通过。input1和input2是要比对的两个图像。c是参考的阈值。支持24位BMP图像。 |
| int LessThan(char\* input1,char\* input2,double c) | 若比对图像的梯度幅相似性偏差值小于c则通过。input1和input2是要比对的两个图像。c是参考的阈值。支持24位BMP图像。 |
| double GMSD(char\* input1, char\* input2) | 求两幅图像的梯度幅相似性偏差值并返回结果。input1和input2是要比对的两个图像。支持24位BMP图像。 |
| void FileWrite(char\* BMP,char\* TXT) | 图像隐写之文件写入，将文本文件写入图像。支持32位BMP图像。BMP是要写入的图像文件名，TXT是要写入图像的文本文件名。 |
| void FileWriteOut(char\* BMP,char\* TXT) | 图像隐写之文件写出，将文本文件从图像中取出来。支持32位BMP图像。BMP是要写出的图像文件名，TXT是写出图像后信息保存的文本文件名。 |
| void Watershed2(char\* input,char\* inputMarqueurs,char\* output,int r,unsigned char R,unsigned char G,unsigned char B) | 图像分割之分水岭算法。inputMarqueurs是输入图像的标记图像。R=230，G=0，B=0，r=1。支持24位BMP图像。 |
| void EcrireImage1(char\* input,char\* output,uint32\_t rayon) | 图像分割。rayon=5。支持24位BMP图像。 |
| void EcrireImage2(char\* input,char\* inputMarqueurs,char\* output,uint32\_t rayon) | 图像分割。rayon=5。支持24位BMP图像。 |
| void EcrireLPECouleur1(char\* input,char\* inputMarqueurs,char\* output,uint32\_t rayon) | 图像分割。rayon=5。支持24位BMP图像。 |
| void Watershed1(char\* input,char\* inputMarqueurs,char\* output,uint32\_t rayon) | 图像分割之分水岭算法。inputMarqueurs是输入图像的标记图像。rayon=5。支持24位BMP图像。 |
| void EcrireImage3(char\* input,char\* inputMarqueurs,char\* output,uint16\_t rayon) | 图像分割。rayon=1。支持24位BMP图像。 |
| void EcrireImageCouleursAleatoires(char\* input,char\* inputMarqueurs,char\* output,uint8\_t r,uint8\_t g,uint8\_t b,uint16\_t rayon) | 图像分割。rayon=1。支持24位BMP图像。 |
| void Watershed(char\* input,char\* inputMarqueurs,char\* output,uint8\_t r,uint8\_t g,uint8\_t b,uint8\_t a,uint16\_t rayon) | 图像分割之分水岭算法。inputMarqueurs是输入图像的标记图像。a一般为255，rayon=1。支持24位BMP图像。 |
| double CharacterRecognition(char\* TargetImage,char\* TemplateFileGroup[]) | 字符匹配，支持BMP图像，返回值是目标图像匹配到的模板文件的序号，如返回值是2则说明图像与序号为2（序号从零开始）的模板匹配。  参考：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[]) | 字符匹配，支持BMP图像，返回值是目标图像匹配到的模板文件的序号，如返回值是2则说明图像与序号为2（序号从零开始）的模板匹配。  参考：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) | 字符分割。支持BMP图像。  OutputFolder是结果输出的文件夹，如“output”，输出结果的文件名的构成方式为：左上角的X坐标-左上角的Y坐标-右下角的X坐标-右下角的Y坐标，YHistogramValleyMaxPixelNumber是求Y方向直方图，谷的最少黑色像素个数，YHistogramValleyMaxPixelNumber=0，XHistogramValleyMaxPixelNumber是求X方向直方图，谷的最少黑色像素个数，XHistogramValleyMaxPixelNumber=4，SubImgBlackPixelPercentage是一张子图内黑色像素超过一定百分比才算有数字，SubImgBlackPixelPercentage=0.001，SingleNumberImgBoundary是单张数字图像边缘填充宽度，SingleNumberImgBoundary=5，Infinite视作无穷大，Infinite=249480，NumberImageBlackPixelPercentage是单张数字图像黑色像素个数超过所有数字图像，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) | 字符分割。支持BMP图像。  BinaryGap是图像二值化全局阈值，BinaryGap=135，BoundaryRemoveGap是边缘全设为白色的距离，BoundaryRemoveGap=7，Infinite是视作无穷大，Infinite=249480，SingleNumberImgBoundary是单张数字图像边缘填充宽度，SingleNumberImgBoundary=5，YHistogramValleyMaxPixelNumber是求Y方向直方图，谷的最少黑色像素个数，YHistogramValleyMaxPixelNumber=0，XHistogramValleyMaxPixelNumber是求X方向直方图，谷的最少黑色像素个数，XHistogramValleyMaxPixelNumber=4，SubImgBlackPixelPercentage是一张子图内黑色像素超过一定百分比才算有数字，SubImgBlackPixelPercentage=0.001，NumberImageBlackPixelPercentage是单张数字图像黑色像素个数超过所有数字图像，NumberImageBlackPixelPercentage=0.35。  参考：output="output"。 |
| void CodeEncoding(std::string input,char\* output, int width,int height, int margin, int eccLevel, int stride\_bytes, int comp,int a) | 二维码编码。input是要编码的字符串，output是生成的二维码图像文件名。  margin:条形码周围的边距  ecc：纠错级别，[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  参考：margin=10，eccLevel=-1，stride\_bytes=0，comp=1。 |
| std::string CodeDecoding(char\* input,int req\_comp,int a) | 二维码解码。input是输入的二维码图像文件名，返回解码结果。  a=1：Lum  a=2：RGB  a=3：BGR  a=4：RGBX  a=5：XRGB  a=6：BGRX  a=7：XBGR  参考：req\_comp=4，a=4。 |