

Manual do utilizador

void ImageFusion(char* input1, char* input2, char* MaskImage, char* output, int dx[], int dy[], int a, double b1, int DX1, int DY1, double EPS)	Fusão de imagens. referência: a=3, b1=4, DX1=-68, DY1=-99, EPS=1, input1=" Fusão de imagens 1. jpg", input2=" Fusão de imagens 2. jpg", MaskImage=" Máscara. png", output="output. jpg". int dx[] = {0,0,-1,1}; int dy[] = {-1,1,0,0};
void Screenshot1(HWND hWnd, LPCWSTR OutputImage)	Função de captura de ecrã. hWnd é a alça da janela para ser captura de tela, como: GetDesktopWindow(); OutputImage é o nome da captura de tela.
void Screenshot2(HWND hWnd, LPCWSTR OutputImage)	Função de captura de ecrã. hWnd é a alça da janela para ser captura de tela, como: GetDesktopWindow(); OutputImage é o nome da captura de tela.
void Screenshot3(HWND hWnd, LPCWSTR OutputImage)	Função de captura de ecrã. hWnd é a alça da janela para ser captura de tela, como: GetDesktopWindow(); OutputImage é o nome da captura de tela.
uint8_t* AESencrypt(uint8_t* input, uint8_t* key, int size)	Função de criptografia AES, onde input é os dados originais, key é a chave e size é o tamanho de input. Devolve os dados encriptados do resultado.
uint8_t* AESdecrypt(uint8_t* input, uint8_t* key, int size)	Função de descriptografia AES, onde input é os dados criptografados, key é a chave, e size é o tamanho de input. Devolve os dados do resultado da descriptografia.
void DES_Encrypt(char *PlainFile, char *Key, char *CipherFile)	Função de encriptação DES, suportando vários ficheiros. PlainFile é o nome do arquivo original, Key é o caractere chave e CipherFile é o nome do arquivo criptografado.
void DES_Decrypt(char *CipherFile, char *Key, char *PlainFile)	DES função de descriptografia, suportando vários arquivos. CipherFile é o nome do arquivo criptografado, Key é o caractere de

	chave e PlainFile é o nome do arquivo descriptografado.
int Equal(char* input1, char* input2, double c)	Se o valor do desvio de similaridade da amplitude do gradiente da imagem comparada for igual a c, ele é passado. Input1 e input2 são duas imagens a serem comparadas. c é o limiar de referência. Suporta imagens BMP de 24 bits.
int GreaterThan(char* input1, char* input2, double c)	Se o valor do desvio de similaridade da amplitude do gradiente da imagem comparada for maior que c, ele é passado. Input1 e input2 são duas imagens a serem comparadas. c é o limiar de referência. Suporta imagens BMP de 24 bits.
int LessThan(char* input1, char* input2, double c)	Se o valor do desvio de similaridade da amplitude do gradiente da imagem comparada for menor que c, ele é passado. Input1 e input2 são duas imagens a serem comparadas. c é o limiar de referência. Suporta imagens BMP de 24 bits.
double GMSD(char* input1, char* input2)	Encontre o valor de desvio de similaridade gradiente entre duas imagens e retorne o resultado. input1 e input2 são duas imagens a serem comparadas. Suporta imagens BMP de 24 bits.
void FileWrite(char* BMP, char* TXT)	Escreva o arquivo de esteganografia da imagem e escreva o arquivo de texto na imagem. Suporta imagens BMP de 32 bits. BMP é o nome do arquivo da imagem a ser escrita, e TXT é o nome do arquivo de texto da imagem a ser escrita.
void FileWriteOut(char* BMP, char* TXT)	Escreva o arquivo de esteganografia da imagem e extraia o arquivo de texto da imagem. Suporta imagens BMP de 32 bits. BMP é o nome do arquivo de imagem a ser escrito, e

	TXT é o nome do arquivo de texto onde as informações são salvas após a gravação da imagem.
void Watershed2(char* input, char* inputMarqueurs, char* output, int r, unsigned char R, unsigned char G, unsigned char B)	0 algoritmo divisor de águas para segmentação de imagens. inputMarqueurs é a imagem anotada da imagem de entrada. R=230, G=0, B=0, r=1. Suporta imagens BMP de 24 bits.
void EcrireImage1(char* input, char* output, uint32_t rayon)	Segmentação de imagens. rayon=5. Suporta imagens BMP de 24 bits.
void EcrireImage2(char* input, char* inputMarqueurs, char* output, uint32_t rayon)	Segmentação de imagens. rayon=5. Suporta imagens BMP de 24 bits.
void EcrireLPECouleur1(char* input, char* inputMarqueurs, char* output, uint32_t rayon)	Segmentação de imagens. rayon=5. Suporta imagens BMP de 24 bits.
void Watershed1(char* input, char* inputMarqueurs, char* output, uint32_t rayon)	0 algoritmo divisor de águas para segmentação de imagens. inputMarqueurs é a imagem anotada da imagem de entrada. rayon=5. Suporta imagens BMP de 24 bits.
void EcrireImage3(char* input, char* inputMarqueurs, char* output, uint16_t rayon)	Segmentação de imagens. rayon=1. Suporta imagens BMP de 24 bits.
void EcrireImageCouleursAleatoires(char* input, char* inputMarqueurs, char* output, uint8_t r, uint8_t g, uint8_t b, uint16_t rayon)	Segmentação de imagens. rayon=1. Suporta imagens BMP de 24 bits.
void Watershed(char* input, char* inputMarqueurs, char* output, uint8_t r, uint8_t g, uint8_t b, uint8_t a, uint16_t rayon)	0 algoritmo divisor de águas para segmentação de imagens. inputMarqueurs é a imagem anotada da imagem de entrada. A é geralmente 255, e rayon=1. Suporta imagens BMP de 24 bits.
double CharacterRecognition(char* TargetImage, char*	Correspondência de caracteres, suporta imagens BMP, e o valor de retorno é o número de sequência do

TemplateFileGroup[])	<p>arquivo de modelo correspondente à imagem de destino. Se o valor de retorno é 2, ele indica que a imagem corresponde ao modelo com o número de sequência 2 (a partir de zero).</p> <p>referência :</p> <pre>TemplateFileGroup[]={ "0.txt", "1.txt", "2.txt", "3.txt", "4.txt", "5.txt", "6.txt", "7.txt", "8.txt", "9.txt" };</pre>
double CharacterRecognition1(char* TargetImage, char* TemplateFileGroup[])	<p>Correspondência de caracteres, suporta imagens BMP, e o valor de retorno é o número de sequência do arquivo de modelo correspondente à imagem de destino. Se o valor de retorno é 2, ele indica que a imagem corresponde ao modelo com o número de sequência 2 (a partir de zero).</p> <p>referência :</p> <pre>TemplateFileGroup[]={ "0.txt", "1.txt", "2.txt", "3.txt", "4.txt", "5.txt", "6.txt", "7.txt", "8.txt", "9.txt" };</pre>
void CharacterSegmentation(char* input, string OutputFolder, int YHistogramValleyMaxPixelNumber, int XHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int SingleNumberImgBoundary, int Infinite, double NumberImageBlackPixelPercentage)	<p>Segmentação de caracteres. Suporta imagens BMP.</p> <p>OutputFolder é a pasta para a saída de resultados, como "output". O formato do nome do arquivo para a saída de resultados é: X coordenada no canto superior esquerdo - Y coordenada no canto superior esquerdo - X coordenada no canto inferior direito - Y coordenada no canto inferior direito , YHistogramValleyMaxPixelNumber é o número mínimo de pixels pretos no vale do histograma de direção Y, YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber é o número mínimo de pixels pretos no vale do histograma de direção X, XHistogramValleyMaxPixelNumber=4,</p>

	<p>SubImgBlackPixelPercentage é a porcentagem de pixels pretos em um subgrafo que é considerado um número ,</p> <p>SubImgBlackPixelPercentage=0.001,</p> <p>SingleNumberImgBoundary é a largura de preenchimento de borda de uma única imagem digital ,</p> <p>SingleNumberImgBoundary=5, Infinito é considerado infinito ,</p> <p>Infinite=249480 ,</p> <p>NumberImageBlackPixelPercentage é o número de pixels pretos em uma única imagem digital que excede todas as imagens digitais ,</p> <p>NumberImageBlackPixelPercentage=0.35.</p>
<pre>void CharacterSegmentation(char* input, char* output, int BoundaryRemoveGap, int BinaryGap, int YHistogramValleyMaxPixelNumber, double SubImgBlackPixelPercentage, int Infinite, int XHistogramValleyMaxPixelNumber, double NumberImageBlackPixelPercentage , int SingleNumberImgBoundary)</pre>	<p>Segmentação de caracteres. Suporta imagens BMP.</p> <p>BinaryGap é o limite global para binarização de imagens ,</p> <p>BinaryGap=135, BoundaryRemoveGap é a distância em que todas as bordas estão definidas para branco ,</p> <p>BoundaryRemoveGap=7 , Infinito é considerado infinito ,</p> <p>Infinite=249480 ,</p> <p>SingleNumberImgBoundary é a largura de preenchimento de borda de uma única imagem digital ,</p> <p>SingleNumberImgBoundary=5 ,</p> <p>YHistogramValleyMaxPixelNumber é o número mínimo de pixels pretos no vale do histograma de direção Y, YHistogramValleyMaxPixelNumber=0,</p> <p>XHistogramValleyMaxPixelNumber é o número mínimo de pixels pretos no vale do histograma de direção X, XHistogramValleyMaxPixelNumber=4,</p> <p>SubImgBlackPixelPercentage é a porcentagem de pixels pretos em um subgrafo que é considerado um número ,</p> <p>SubImgBlackPixelPercentage=0.001,</p>

	<p>NumberImageBlackPixelPercentage é o número de pixels pretos em uma única imagem digital que excede todas as imagens digitais , NumberImageBlackPixelPercentage=0.35.</p> <p>Referência: output="output".</p>
<pre>void CodeEncoding(std::string input, char* output, int width, int height, int margin, int eccLevel, int stride_bytes, int comp, int a)</pre>	<p>Codificação de código QR. Entrada é a string a ser codificada, e saída é o nome do arquivo da imagem gerada do código QR.</p> <p>margem: A margem em torno do código de barras</p> <p>ecc: Nível de correcção de erros, [0-8]</p> <p>a=1: AZTEC</p> <p>a=2: CODABAR</p> <p>a=3: CODE_39</p> <p>a=4: CODE_93</p> <p>a=5: CODE_128</p> <p>a=6: DATA_MATRIX</p> <p>a=7: EAN_8</p> <p>a=8: EAN_13</p> <p>a=9: ITF</p> <p>a=10: MAXICODE</p> <p>a=11: PDF_417</p> <p>a=12: QR_CODE</p> <p>a=13: RSS_14</p> <p>a=14: RSS_EXPANDED</p> <p>a=15: UPC_A</p> <p>a=16: UPC_E</p> <p>a=17: UPC_EAN_EXTENSION</p> <p>Referência: margin=10, eccLevel=-1, stride_bytes=0, comp=1.</p>
<pre>std::string CodeDecoding(char* input, int req_comp, int a)</pre>	<p>Descodificação de código QR. Entrada é o nome do arquivo da imagem de código QR de entrada e retorna o resultado de decodificação.</p> <p>a=1: Lum</p> <p>a=2: RGB</p> <p>a=3: BGR</p> <p>a=4: RGBX</p> <p>a=5: XRGB</p>

	<p>a=6: BGRX</p> <p>a=7: XBGR</p> <p>Referência: req_comp=4, a=4。</p>
--	---