Instrucciones de uso

void ImageFusion(char* input1,char* input2,char* MaskImage,char* output,int dx[],int dy[],int a,double b1,int DX1,int DY1,double EPS)	Fusión de imágenes. Referencia: a=3, b1=4, DX1=-68, DY1=-99, EPS=1, input1="Fusión de imágenes 1.jpg", input2="Fusión de imágenes 2.jpg", MaskImage="Mascarilla.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)	Función de cifrado DES, compatible con una variedad de archivos. PlainFile es el nombre del archivo original, Key es el carácter clave y CipherFile es el nombre del archivo cifrado.
void DES_Decrypt(char *CipherFile, char *Key,char *PlainFile)	La función de descifrado des admite una variedad de archivos. CipherFile es el nombre del archivo encriptado, Key es el carácter clave y PlainFile es el nombre del archivo descifrado.
void FileWrite(char* BMP,char* TXT)	Se escribe un archivo oculto de la imagen, y se escribe un archivo de texto en la imagen. Admite imágenes BMP de 32 bits. BMP es el nombre del archivo de imagen a escribir, y txt es el nombre del archivo de texto a escribir en la imagen.
void FileWriteOut(char* BMP,char* TXT)	Se escribe un archivo oculto de la imagen, que extrae el archivo de texto de la imagen. Admite imágenes BMP de 32 bits. BMP es el nombre del archivo de imagen a escribir, y txt es el nombre del archivo de texto guardado por la información después de escribir la imagen.
void Watershed2(char* input,char* inputMarqueurs,char* output,int r,unsigned char R,unsigned char G,unsigned char B)	Algoritmo de cuenca para la segmentación de imágenes. inputMarqueurs es una imagen marcada de la imagen introducida. R = 230, G = 0, B = 0, r = 1. Admite imágenes BMP de 24 bits.
void EcrireImage1(char* input,char* output,uint32_t	Segmentación de imágenes. rayon = 5. Admite imágenes PNG.

rayon)	
void EcrireImage2(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Segmentación de imágenes. rayon = 5. Admite imágenes PNG.
void EcrireLPECouleur1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Segmentación de imágenes. rayon = 5. Admite imágenes PNG.
void Watershed1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Algoritmo de cuenca para la segmentación de imágenes. inputMarqueurs es una imagen marcada de la imagen introducida. rayon = 5. Admite imágenes PNG.
void EcrireImage3(char* input,char* inputMarqueurs,char* output,uint16_t rayon)	Segmentación de imágenes. rayon = 1. Admite imágenes PNG.
void EcrireImageCouleursAleatoires(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint16_t rayon)	Segmentación de imágenes. rayon = 1. Admite imágenes PNG.
void Watershed(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint8_t a,uint16_t rayon)	Algoritmo de cuenca para la segmentación de imágenes. Inputmarqueurs es una imagen marcada de la imagen introducida. a generalmente es 255, rayon = 1. Admite imágenes PNG.
double CharacterRecognition(char* TargetImage,char* TemplateFileGroup[])	Coincidencia de caracteres, soporte para imágenes bmp, el valor de retorno es el número de serie del archivo de plantilla al que coincide la imagen objetivo, si el valor de retorno es 2, significa que la imagen coincide con la plantilla con el número de serie 2 (el número de serie comienza en cero). Referencia : 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[])	Coincidencia de caracteres, soporte para imágenes bmp, el valor de retorno es el número de serie del archivo de plantilla al que coincide la imagen objetivo, si el valor de retorno es 2, significa que la imagen coincide con la plantilla con el número de serie 2 (el número de serie comienza en cero). Referencia : 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)	División de caracteres. Admite imágenes bmp. OutputFolder es la carpeta a la que se exportan los resultados, como "output", y el nombre del archivo que exporta los resultados se compone de: la coordenada X en la esquina superior izquierda - la coordenada y en la esquina superior izquierda - la coordenada X en la esquina inferior derecha - la coordenada y en la esquina inferior derecha , YHistogramValleyMaxPixelNumber es el número mínimo de píxeles negros en el valle para encontrar el histograma de dirección Y , YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber es encontrar el histograma de dirección x, el número mínimo de píxeles negros en el valle , XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage es un subinforme en el que los píxeles negros superan un cierto porcentaje para tener números , SubImgBlackPixelPercentage=0.001,

	SingleNumberImgBoundary es el ancho de relleno del borde de una sola imagen digital , SingleNumberImgBoundary=5,Infinite considera infinito , Infinite=249480 , NumberImageBlackPixelPercentage es una sola imagen digital con más píxeles negros que todas las imágenes digitales , 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)	División de caracteres. Admite imágenes bmp. BinaryGap es el umbral global de la binarización de la imagen , BinaryGap=135,BoundaryRemoveGap es una distancia en la que los bordes están todos establecidos en blanco, BoundaryRemoveGap=7 , Infinite se considera infinito , Infinite=249480 , SingleNumberImgBoundary es el ancho de relleno del borde de una sola imagen digital , SingleNumberImgBoundary=5 , YHistogramValleyMaxPixelNumber es el número mínimo de píxeles negros en el valle para encontrar el histograma de dirección Y , YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber es encontrar el histograma de dirección x, el número mínimo de píxeles negros en el valle , XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage es un subinforme en el que los píxeles negros superan un cierto porcentaje para tener números , SubImgBlackPixelPercentage=0.001, NumberImageBlackPixelPercentage es una sola imagen digital con más píxeles negros que todas las

	imágenes digitales , NumberImageBlackPixelPercentage=0.35。 Referencia: output="output"。
void CodeEncoding(std::string input,char* output, int width,int height, int margin, int eccLevel, int stride_bytes, int comp,int a)	Codificación de código qr. input es la cadena a codificar y output es el nombre del archivo de imagen de código QR generado. margin: márgenes alrededor del Código de barras ecc: nivel de corrección de errores, [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)	Decodificación de código qr. input es el nombre de archivo de imagen de código QR introducido, que devuelve el resultado de la decodificación. 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。