Manuel d'utilisation

void ImageFusion(char* input1,char* input2,char* MaskImage,char* output,int dx[],int dy[],int a,double b1,int DX1,int DY1,double EPS)	Fusion d'images. Référence: a=3, b1=4, DX1=-68, DY1=-99, EPS=1, input1="Fusion d'images1.jpg", input2="Fusion d'images2.jpg", MaskImage="Le masque.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)	Fonction de chiffrement des, prend en charge plusieurs types de fichiers. PlainFile est le nom de fichier du fichier original, Key est le caractère clé et CipherFile est le nom de fichier chiffré.
void DES_Decrypt(char *CipherFile, char *Key,char *PlainFile)	Fonction de décryptage des, support de plusieurs types de fichiers. CipherFile est le nom de fichier du fichier chiffré, Key est le caractère clé et PlainFile est le nom de fichier après déchiffrement.
void FileWrite(char* BMP,char* TXT)	écriture de fichier en écriture implicite d'image, écriture de fichier texte dans l'image. Prise en charge des images BMP 32 bits. BMP est le nom du fichier image à écrire et txt est le nom du fichier texte à écrire dans l'image.
void FileWriteOut(char* BMP,char* TXT)	Écrire le fichier d'écriture implicite de l'image pour extraire le fichier texte de l'image. Prise en charge des images BMP 32 bits. BMP est le nom du fichier image à écrire et txt est le nom du fichier texte dans lequel les informations sont sauvegardées après l'écriture de l'image.
void Watershed2(char* input,char* inputMarqueurs,char* output,int r,unsigned char R,unsigned char G,unsigned char B)	Algorithme de division de l'image. Inputmarqueurs est une image marquée de l'image d'entrée. R = 230, G = 0, B = 0, r = 1. Prend en charge les images BMP 24 bits.
void EcrireImage1(char* input,char* output,uint32_t	Segmentation de l'image. rayon = 5. Prend en charge les images PNG.

rayon)	
void EcrireImage2(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Segmentation de l'image. rayon = 5. Prend en charge les images PNG.
void EcrireLPECouleur1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Segmentation de l'image. rayon = 5. Prend en charge les images PNG.
void Watershed1(char* input,char* inputMarqueurs,char* output,uint32_t rayon)	Algorithme de division de l'image. Inputmarqueurs est une image marquée de l'image d'entrée. rayon = 5. Prend en charge les images PNG.
void EcrireImage3(char* input,char* inputMarqueurs,char* output,uint16_t rayon)	Segmentation de l'image. rayon = 1. Supporte les images PNG.
void EcrireImageCouleursAleatoires(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint16_t rayon)	Segmentation de l'image. rayon = 1. Supporte les images PNG.
void Watershed(char* input,char* inputMarqueurs,char* output,uint8_t r,uint8_t g,uint8_t b,uint8_t a,uint16_t rayon)	Algorithme de division de l'image. Inputmarqueurs est une image marquée de l'image d'entrée. a est généralement 255 et rayon = 1. Supporte les images PNG.
double CharacterRecognition(char* TargetImage,char* TemplateFileGroup[])	Correspondance de caractères, prise en charge des images BMP, la valeur de retour est le numéro de série du fichier modèle auquel l'image cible correspond, si la valeur de retour est 2, l'image correspond au modèle dont le numéro de série est 2 (le numéro de série commence à zéro). Référence : 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[])	Correspondance de caractères, prise en charge des images BMP, la valeur de retour est le numéro de série du fichier modèle auquel l'image cible correspond, si la valeur de retour est 2, l'image correspond au modèle dont le numéro de série est 2 (le numéro de série commence à zéro). Référence : 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)	Segmentation des caractères. Supporte les images BMP. OutputFolder est le dossier de sortie du résultat, tel que "output", le nom du fichier de sortie est constitué de la manière suivante: coordonnées x en haut à gauche - coordonnées y en haut à gauche - coordonnées x en bas à droite - coordonnées y en bas à droite, YHistogramValleyMaxPixelNumber est un histogramme de direction y, nombre minimum de pixels noirs de la vallée , YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber est un histogramme indiquant la direction X, le nombre minimum de pixels noirs de la vallée , XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage est un sous - graphique avec plus d'un certain pourcentage de pixels noirs pour compter les chiffres , SubImgBlackPixelPercentage=0.001, SingleNumberImgBoundary est une image numérique unique bordure remplissage largeur , SingleNumberImgBoundary=5,Infinite

	comme l'infini, Infinite=249480, NumberImageBlackPixelPercentage est une seule image numérique noir nombre de pixels plus que toutes les images numériques , 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)	Segmentation des caractères. Supporte les images BMP. BinaryGap est le seuil global de binarisation d'image , BinaryGap=135 , BoundaryRemoveGap est la distance où les bords sont entièrement mis en blanc , BoundaryRemoveGap=7, Infinite est considéré comme l'infini , Infinite=249480 , SingleNumberImgBoundary est une image numérique unique bordure remplissage largeur , SingleNumberImgBoundary=5 , YHistogramValleyMaxPixelNumber est un histogramme indiquant la direction y, le nombre minimum de pixels noirs de la vallée , YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber est un histogramme indiquant la direction X, le nombre minimum de pixels noirs de la vallée , XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage est un sous - graphique avec plus d'un certain pourcentage de pixels noirs pour compter les chiffres , SubImgBlackPixelPercentage=0.001, NumberImageBlackPixelPercentage est une seule image numérique noir nombre de pixels plus que toutes les images numériques , NumberImageBlackPixelPercentage=0.35。 Référence: output="output"。
void CodeEncoding(std::string	Code 2D codé. Input est la chaîne

input,char* output, int width,int height, int margin, int eccLevel, int stride_bytes, int comp,int a)	à encoder et Output est le nom du fichier image du Code QR généré. margin: marges autour du Code à barres eccLevel: niveau de correction d'erreur, [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 Référence: margin=10, eccLevel=-1, stride_bytes=0, comp=1。
std::string CodeDecoding(char* input,int req_comp,int a)	Décodage du code 2D. Input est le nom du fichier image du Code QR entré qui renvoie le résultat du décodage. a=1: Lum a=2: RGB a=3: BGR a=4: RGBX a=5: XRGB a=6: BGRX a=7: XBGR Référence: req_comp=4, a=4。