Manuel d'utilisation

void ImageFusion(char*	Fusion d'images multifocales avec
input1, char* input2, char*	prise en charge des images BMP 8
output, int block height, int	bits. block_height = 8, block_width
block_width, double threshold)	= 8, threshold = 1,75.
void ImageFusion(char*	Fusion d'images. Référence: a=3,
input1, char* input2, char*	b1=4, DX1=-68, DY1=-99, EPS=1,
MaskImage, char* output, int	input1=" Fusion d'images1.jpg",
dx[], int $dy[]$, int a, double	input2=" Fusion d'images2.jpg",
b1, int DX1, int DY1, double EPS)	MaskImage=" Le masque.png",
2, 110 2111, 2110 2111, 0000 20 21 27	output="output.jpg"。
	$int dx[] = \{0, 0, -1, 1\};$
	int $dy[] = \{-1, 1, 0, 0\};$
void ImageFusion(char*	Fusion d'image, supporte les images
input1, char* input2, char*	PNG. Référence: input1=" Fusion
inputUniqe1, char*	d'images1.png", input2=" Fusion
inputUniqe2, char* output)	d'images2.png", inputUniqe1="
impationique, chai · output/	Fusion d'imagesl_unique.txt",
	inputUniqe2="Fusion
	d'images2_unique.txt".
void Uniqe(char* input, char*	Fusion d'image, supporte les images
inputUniqe, char* output, double	PNG. Référence: input=" Fusion
R, double G, double B)	d'images1.png", inputUniqe="
in, double of double by	Fusion d'imagesl_unique.txt".
	R=255, G=0, B=0.
void Screenshot1(HWND hWnd,	Fonction de capture d'écran. hWnd
LPCWSTR OutputImage)	est le Handle de la fenêtre à
	capturer, comme :
	GetDesktopWindow(); Outputimage est
	le nom de la capture d'écran.
void Screenshot2(HWND	Fonction de capture d'écran. hWnd
hWnd, LPCWSTR OutputImage)	est le Handle de la fenêtre à
	capturer, comme :
	GetDesktopWindow(); Outputimage est
	le nom de la capture d'écran.
void Screenshot3(HWND hWnd,	Fonction de capture d'écran. hWnd
LPCWSTR OutputImage)	est le Handle de la fenêtre à
	capturer, comme :
	GetDesktopWindow(); Outputimage est
	le nom de la capture d'écran.
uint8_t* AESencrypt(uint8_t*	Fonction de chiffrement AES, input
input, uint8_t* key, int size)	est la donnée brute, key est la clé
	et size est la taille de input.
	Retourne les données de résultat

	chiffrées.
uint8 t* AESdecrypt(uint8 t*	Fonction de déchiffrement AES,
input, uint8_t* key, int size)	input est la donnée chiffrée, key
	est la clé et size est la taille de
	input. Retourne les données de
	résultat déchiffrées.
void DES_Encrypt(char	Fonction de chiffrement des, prend
*PlainFile, char *Key,char	en charge plusieurs types de
*CipherFile)	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	Fonction de décryptage des, support
*CipherFile, char *Key, char	de plusieurs types de fichiers.
*PlainFile)	CipherFile est le nom de fichier du
,	fichier chiffré, Key est le
	caractère clé et PlainFile est le
	nom de fichier après déchiffrement.
int Equal(char* input1, char*	Passe si la valeur de l'écart de
input2, double c)	similarité de l'amplitude de
imputz, double c)	gradient de l'image alignée est
	égale à c. Input1 et input2 sont
	les deux images à aligner. c est le
	seuil de référence. Prend en charge
	les images BMP 24 bits.
int GreaterThan(char*	Passe si la valeur de l'écart de
input1, char* input2, double c)	similarité d'amplitude de gradient
	de l'image alignée est supérieure
	à c. Input1 et input2 sont les deux
	images à aligner, c est le seuil de
	référence. Prend en charge les
	images BMP 24 bits.
int LessThan(char* input1, char*	Passe si la valeur de l'écart de
input2, double c)	similitude d'amplitude de gradient
	de l'image alignée est inférieure
	à c. Input1 et input2 sont les deux
	images à aligner. c est le seuil de
	référence. Prend en charge les
	images BMP 24 bits.
double GMSD(char* input1, char*	Déterminez la valeur de l'écart de
input2)	similitude d'amplitude de gradient
111p 0 0 0 0 1	pour les deux images et retournez
	le résultat. input1 et input2 sont
	les deux images à aligner. Prend en

	1 . 1 . DWD 04.1.
11 D11 W 11 (1 1 DVD 1	charge les images BMP 24 bits.
void FileWrite(char* BMP, char*	écriture de fichier en écriture
TXT)	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*	Écrire le fichier d'écriture
BMP, char* TXT)	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*	Algorithme de division de l'image.
input, char*	Inputmarqueurs est une image
inputMarqueurs, char* output, int	marquée de l'image d'entrée. R =
r, unsigned char R, unsigned char	230, G = 0, B = 0, r = 1. Prend en
G, unsigned char B)	charge les images BMP 24 bits.
void EcrireImage1(char*	Segmentation de l'image. rayon = 5.
input, char* output, uint32_t	Prend en charge les images BMP 24
rayon)	bits.
void EcrireImage2(char*	Segmentation de l'image. rayon = 5.
input, char*	Prend en charge les images BMP 24
inputMarqueurs, char*	bits.
output, uint32_t rayon)	
void EcrireLPECouleur1(char*	Segmentation de l'image. rayon = 5.
input, char*	Prend en charge les images BMP 24
inputMarqueurs, char*	bits.
output, uint32_t rayon)	
void Watershed1(char*	Algorithme de division de l'image.
input, char*	Inputmarqueurs est une image
inputMarqueurs, char*	marquée de l'image d'entrée. rayon
output, uint32_t rayon)	= 5. Prend en charge les images BMP
	24 bits.
void EcrireImage3(char*	Segmentation de l'image. rayon = 1.
input, char*	Prend en charge les images BMP 24
inputMarqueurs, char*	bits.
output, uint16_t rayon)	
void	Segmentation de l'image. rayon = 1.
EcrireImageCouleursAleatoires(c	Prend en charge les images BMP 24

har* input, char*	bits.
inputMarqueurs, char*	51 (3.
output, uint8_t r, uint8_t	
g, uint8 t b, uint16 t rayon)	
void Watershed (char*	Algorithme de division de l'image.
input, char*	inputMarqueurs est une image
inputMarqueurs, char*	marquée de l'image d'entrée.
output, uint8_t r, uint8_t	Généralement 255, rayon = 1. Prend
g, uint8_t b, uint8_t a, uint16_t	en charge les images BMP 24 bits.
rayon) double	Correspondance de caractères,
	-
CharacterRecognition(char*	prise en charge des images BMP, la
TargetImage, char*	valeur de retour est le numéro de
TemplateFileGroup[])	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 :
	<pre>TemplateFileGroup[]={</pre>
	1. txt ,
	"4. txt", "5. txt", "6. txt",
	"7. txt", "8. txt", "9. txt" };
double	Correspondance de caractères,
CharacterRecognition1(char*	prise en charge des images BMP, la
TargetImage, char*	valeur de retour est le numéro de
TemplateFileGroup[])	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",
. 1	"7. txt", "8. txt", "9. txt" };
void	Segmentation des caractères.
CharacterSegmentation(char*	Supporte les images BMP.
input, string OutputFolder, int	OutputFolder est le dossier de
YHistogramValleyMaxPixelNumber,	sortie du résultat, tel que
int	"output", le nom du fichier de
XHistogramValleyMaxPixelNumber,	sortie est constitué de la manière

double
SubImgBlackPixelPercentage, int
SingleNumberImgBoundary, int
Infinite, double
NumberImageBlackPixelPercentage
)

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 nombre minimum de pixels noirs de 1a val1ée YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber est histogramme indiquant direction X, le nombre minimum de pixels noirs de 1a vallée, XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage est un sous - graphique avec plus d'un certain pourcentage de pixels noirs 1es pour compter chiffres , SubImgBlackPixelPercentage=0.001, SingleNumberImgBoundary image numérique unique bordure remplissage 1argeur SingleNumberImgBoundary=5, Infinite comme l'infini, Infinite=249480, NumberImageBlackPixelPercentage est une seule image numérique noir nombre de pixels plus que toutes 1es 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 blanc en BoundaryRemoveGap=7, Infinite est considéré comme l'infini Infinite=249480 SingleNumberImgBoundary une image numérique unique bordure remplissage largeur

SingleNumberImgBoundary=5 YHistogramValleyMaxPixelNumber est histogramme indiquant direction y, le nombre minimum de pixels noirs de 1a vallée, YHistogramValleyMaxPixelNumber=0, XHistogramValleyMaxPixelNumber est histogramme indiquant direction X, le nombre minimum de pixels noirs de 1a vallée, XHistogramValleyMaxPixelNumber=4, SubImgBlackPixelPercentage est un sous - graphique avec plus d'un certain pourcentage de pixels noirs 1es chiffres , pour compter SubImgBlackPixelPercentage=0.001, NumberImageBlackPixelPercentage est une seule image numérique noir nombre de pixels plus que toutes 1es images numériques NumberImageBlackPixelPercentage=0. 35。 Référence: output="output".

void CodeEncoding(std::string
input, char* output, int
width, int height, int margin,
int eccLevel, int stride_bytes,
int comp, int a)

Code 2D codé. Input est la chaîne 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*	Décodage du code 2D. Input est le
<pre>input, int req_comp, int a)</pre>	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.