使用说明书

目录

PPM、PGM 和 PBM 图像处理
YUV 图像处理
RAW 图像处理
3MP 图像处理
其它处理
高级算子

PPM、PGM 和 PBM 图像处理

void OTSUBinarization(char* input, char* output) void PPMtoBMP(char* input, char* output, int bpp) void BMPtoPPM(char* input, char* input, char* output, int bpp) void BMPtoPPM(char* input, char* bMP 图像转 PPM 图像转 PPM 图像。 BMP 图像转 PPM 图像。
PGM 图像。 void PPMtoBMP(char* input, char* PPM 图像转为 BMP 图像。bpp 是 BMP 图如tput, int bpp) 像的色深。 void BMPtoPPM(char* input, char* BMP 图像转 PPM 图像。
void PPMtoBMP(char* input, char* output, int bpp)PPM 图像转为 BMP 图像。bpp 是 BMP 图像的色深。void BMPtoPPM(char* input, char* BMP 图像转 PPM 图像。
output, int bpp)像的色深。void BMPtoPPM(char* input, char* BMP 图像转 PPM 图像。
void BMPtoPPM(char* input, char* BMP 图像转 PPM 图像。
output)
void PPMtoBMP1(char* input, char* PPM 图像转为 BMP 图像。bpp 是 BMP 图
output, int bpp) 像的色深。
void BMPtoPPM1(char* input, char* BMP 图像转 PPM 图像。
output)
void BMPtoPGM(char* input, char* BMP 转 PGM。
output)
void BMPtoPPM2(char* input, BMP转PPM。
char* output)
void PPMtoPGM(char* input, char* PPM 转 PGM。
output)
void BlurPPM(char* input, char* PPM 图像滤波。
output)
void BlurPGM(char* input, char* PGM 图像滤波。
output)
void OTSU二值化划分。input 是输入文件名,
SegmentsOTSUBinarization(char* output 是输出文件名。支持 P5 格式的
input, char* output) PGM 图像。
void P3PPMB1ur(char* input, char* PPM 图像模糊, input 是输入文件名,
output) output 是输出文件名。支持 P3 格式的
PPM 图像。
unsigned char** ReadPBM(char* 读取 PBM 图像并返回图像数据。input
input) 是要读取的 PBM 图像文件名。支持 P4

	格式的 PBM 图像。
void WritePBM(unsigned char**	保存 PBM 图像。Input 是输入的图像数
Input, char* output)	据, output 是输出文件名。支持 P4 格
	式的 PBM 图像。
void	直方图均衡化,input 是输入文件名,
PGMHistogramEqualization(char*	output 是输出文件名。支持 P5 格式的
input, char* output)	PGM 图像。
PPMImage* ReadPPM(char* input)	PPM 图像读取, input 是要读取的 PPM 图像文件名。支持 P6 格式的 PPM 图像。 需要引入的结构体:
	typedef struct {
	unsigned char red, green, blue; //像素的颜色由 RGB(红/绿/
	蓝)表示
	<pre>} PPMPixe1;</pre>
	typedef struct {
	unsigned int width, height;
	// 图像的宽度和高度(以像素 为单位)
	PPMPixel *data;
	// 构成图像的像素
	} PPMImage;
void WritePPM(char* output, PPMImage* img)	PPM 图像保存, output 是输出的 PPM 图像文件名,img 是输入的图像数据。 支持 P6 格式的 PPM 图像。 需要引入的结构体: typedef struct { unsigned char red, green, blue; //像素的颜色由 RGB (红/绿/蓝)表示 } PPMPixel; typedef struct { unsigned int width, height; // 图像的宽度和高度(以像素为单位) PPMPixel *data; // 构成图像的像素 } PPMImage;
void InvertColor(char*	负滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 P6 格式的 PPM 图像。
void GrayFilter(char*	灰度过滤器, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P6 格式的
	PPM 图像。

void SepiaFilter(char*	乌贼墨过滤器,input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P6 格式的
Thiput, char * output)	
	PPM 图像。
void AdjustSaturation(char*	调整图像饱和度,input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标饱和度,
	如 a=30。支持 P6 格式的 PPM 图像。
void Resize(char* input, char*	调整图像大小,input 是输入文件名,
output, unsigned int NewWidth,	output 是输出文件名。NewWidth 和
unsigned int NewHeight)	NewHeight 分别是输出图像的宽和高。
	支持 P6 格式的 PPM 图像。
void AdjustHue(char* input, char*	调整图像的色调, input 是输入文件名,
output, int a)	output 是输出文件名。a 是目标色调,
output, int a)	如 a=125。支持 P6 格式的 PPM 图像。
11 A11. AD. 11. (1 at.	
void AdjustBrightness(char*	调整图像亮度,input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标亮度,
	如 a=60。支持 P6 格式的 PPM 图像。
void AdjustContrast(char*	调整图像对比度, input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标对比度,
	如 a=60。支持 P6 格式的 PPM 图像。
void AdjustBlur(char*	通过 sigma 因子模糊图像, input 是输
input, char* output, double a)	入文件名, output 是输出文件名。a 是
	sigma 因子,如 a=5。支持 P6 格式的 PPM
	图像。
void MeanGrayFilter(char*	平均灰度滤波器,input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是平均值系数,
	如 a=3。支持 P6 格式的 PPM 图像。
void Pixelate(char* input, char*	像素化, input 是输入文件名, output
output, unsigned int a)	是输出文件名。a 是幅度值,如 a=8。
output, unsigned int a)	支持 P6 格式的 PPM 图像。
usid Detete (sharak input sharak	
void Rotate(char* input, char*	旋转图像,input 是输入文件名,output
output, short a)	是输出文件名。a 是旋转的角度,如
,	a=45。支持 P6 格式的 PPM 图像。
void GammaCorrection(char*	伽马校正, input 是输入文件名, output
input, char* output, double a)	是输出文件名。a 是 gamma 数,如 a=0.5。
	支持 P6 格式的 PPM 图像。
void	生成灰度图以及 RGB 通道分离, input
GrayAndChannelSeparation(char*	是输入的 P6 格式的 PPM 图像;
input, char* Grayoutput, char*	Grayoutput 是输出的灰度图文件名,
Routput, char* Goutput, char*	Routput、Goutput 和 Boutput 分别是输
Boutput)	出的 R、G 和 B 通道的图像文件名,输
	出都是 PGM 格式文件。
void PGMBin(char* input, char*	灰度图像二值化,输入是灰度图像,输
	入和输出都是 PGM 文件, threshold 是
output, int threshold)	
	阈值,如 threshold=125。

void Brightening(char*	彩色图像增亮,输入和输出都是 P6 格
input, char* output, int a)	式的 PPM 图像, a 是增亮系数, 如 a=80。
void GrayBrightening(char*	灰度图像增亮,输入和输出都是 PGM 图
input, char* output, int a)	像,a是增亮系数,如 a=80。
void PPMFilter(char* input, char*	彩色图像滤波,输入和输出都是 P6 格
output)	式的 PPM 文件。
void PGMGrayFilter(char*	灰度图像滤波,输入和输出都是 PGM 图
input, char* output)	像。
void PPMtoBMP(char* input, char*	PPM 图像转 BMP 图像, input 是输入文
output)	件名, output 是输出文件名。支持 P6
	格式的 PPM 图像。
void PGMOtsuThreshold(string	大津阈值法, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P5 格式的
	PGM 图像。
void PGMRotated(char*	channels 是输入图像的通道, theta 是
input, char* output, int width, int	旋转弧度,参考:
height, int channels, double	theta=45.0*3.1415926/180。
theta)	
void XCorner(char* input, char*	channels 是输入图像的通道。
output, int width, int height, int	
channels, double theta)	
void YCorner(char* input, char*	channels 是输入图像的通道。
output, int width, int height, int	
channels, double theta)	
void Smooth(char* input, char*	channels 是输入图像的通道,sigma_x
output, int width, int height, int	是X方向的模糊系数,sigma_y是Y方
channels, float sigma_x, float	向的模糊系数。
sigma_y, double theta)	
void	局部大津阈值,input 是输入文件名,
PGMLocalisedOtsuThreshold(string	output 是输出文件名。支持 P5 格式的
input, char* output)	PGM 图像。
void PGMSauvolaThreshold(string	索沃拉阈值,支持 P5 格式的 PGM 图像。
input, char* output, double	a、b和c的参考值如:
a, double b, double c)	a=0.01, b=15, c=225。
void PGMThreshold(string	阈值法,input 是输入文件名,output
input, char* output, int thresh)	是输出文件名。支持 P5 格式的 PGM 图
floot Donoin1/showt 1	像。thresh是阈值,如:thresh=5。
float Repair1(char* input, char*	图像修复,var 是噪声方差,threshold
output, float var, float	是阈值,nbLevels 是要处理的级别数, a=10。返回 ISNR。
threshold, int nbLevels, float a)	图像修复, var 是噪声方差, threshold
float Repair2(char* input, char*	图像修复,var 是噪声力差,threshold 是阈值,nbLevels 是要处理的级别数,
output, float var, float threshold, int nbLevels, float a)	定國恒,IDLEVEIS 定安处垤的级剂数, a=10。返回 ISNR。
void LowPassFilterRepair1(char*	低 通 滤 波 图 像 修 复 , a=10 , b=6 ,

input, char* output, int	nbLevels=3, size_filter 是低通过滤
size filter, float var, int	器的大小, var 是噪声方差,
nb_iterations, int nbLevels, float	nb iterations 是 Landweber 的迭代
a, int b)	数。
void LowPassFilterRepair2(char*	低通滤波图像修复, a=10, b=6,
input, char* output, int	nbLevels=3, size filter 是低通过滤
size_filter, float var, int	器的大小,var是噪声方差,
nb_iterations, int nbLevels, float	nb_iterations 是 Landweber 的迭代
a, int b)	数。
float LowPassFilterRepair3(char*	(低通滤波图像修复, a=10, b=6,
input, char* output, int	ND
	通过滤器的大小, var 是噪声方差,
size_filter, float var, int	DD Carter
nb_iterations, int nbLevels, int	数。返回 ISNR。
pas, float a, int b)	
void Repair1(char* input, char*	图像修复,a=0.0,M 是分解的层次数, m M-2
output, int M, float a)	如 M=3。
void Repair2(char* input, char*	图像修复,a=0.0, M 是分解的层次数,
output, int M, float a)	如 M=3。
void MakeNoisel(char*	制造噪声,size_filter 是低通滤波器
input, char* output, int	的宽度。
size_filter)	
void MakeNoise2(char*	制造噪声,nb_iterations是Landweber
input, char* output, int	的迭代数,pas=1。
nb_iterations, int pas)	
void MakeNoise3(char* output, int	制造噪声, height 是输出图像的高,
height, int width, float var)	width 是输出图像的宽, var 是噪声方
	差。
void MakeNoise4(char*	制造噪声,nb_iterations是Landweber
input, char* output, int	的迭代数,pas=1。
nb_iterations, int pas)	
void ImageReconstruction(char*	图像重建,支持 PGM 文件。参考:
input, char* output, int	maxDepth=80, threshold=50, tx=0,
maxDepth, int threshold, int	ty=0.
tx, int ty)	

YUV 图像处理

<pre>void YUVsuperposition(char* input1, char*</pre>	YUV420 叠加, Y_BLACK、
input2, char* output, int width, int	U_BLACK 和 V_BLACK 用于将
height, unsigned char Y_BLACK, unsigned char	原图中的黑色变成透明,参
U_BLACK, unsigned char V_BLACK)	考 : Y_BLACK=16 ,
	U_BLACK=128 ,
	V_BLACK=128。
void YUVsuperposition(char* input1, char*	YUV444 叠加, Y_BLACK、
input2, char* output, int width, int	U_BLACK 和 V_BLACK 用于将

height, unsigned char Y_BLACK, unsigned char U_BLACK, unsigned char V_BLACK)	原图中的黑色变成透明,参 考 : Y_BLACK=16 , U_BLACK=128 ,
void YUVsuperposition(char* input1, char* input2, char* output, int width, int height, unsigned char Y_BLACK, unsigned char U_BLACK, unsigned char V_BLACK)	V_BLACK=128。 yuv444p 直接叠加到 yuv420p上,不做转换, Y_BLACK、U_BLACK 和 V_BLACK用于将原图中的黑色变成透明,参考: Y BLACK=16, U BLACK=128,
void YUV444toYUV420(char* input, char* output, int height, int width)	V_BLACK=128。 YUV444 转 YUV420, height 是输入的YUV444文件的高, width 是输入的 YUV444文件的宽。
void YUV444toYUV420(char* input, char* output, int height, int width, int frames)	YUV444 转 YUV420, height 和width是输入文件的高和 宽,frames 是要输入文件中 操作的帧序号。
void YUVsuperposition(char* input1, char* input2, char* output, int width, int height, unsigned char Y_BLACK, unsigned char U_BLACK, unsigned char V_BLACK)	YUV444 转到 YUV420 上的叠加, Y_BLACK、U_BLACK和V_BLACK用于将原图中的黑色变成透明,参考:Y_BLACK=16,U_BLACK=128,VBLACK=128。
<pre>void YUVEdgeProcessingY(char* input, char* output, int width, int height, double k)</pre>	YUV 边缘处理, input 是输入文件名, output 是输出文件名。width 和 height 是输入图像的宽和高。参考: k=0.5。
<pre>void YUVEdgeProcessingU(char* input, char* output, int width, int height, double k)</pre>	YUV 边缘处理, input 是输入文件名, output 是输出文件名。width 和 height 是输入图像的宽和高。参考: k=0.5。
<pre>void YUVEdgeProcessingV(char* input, char* output, int width, int height, double k)</pre>	YUV 边缘处理, input 是输入文件名, output 是输出文件名。width 和 height 是输入图像的宽和高。参考: k=0.5。
<pre>void BMPLoadedIntoYUV(char* inputBMP, char* inputYUV, char* output, int YUVwidth, int YUVheight, int depth, bool mt)</pre>	YUV 加载 BMP, inputBMP 是输入的 BMP 图像, inputYUV 是输入的 YUV 图像, inputYUV 起到容器的作用,

void YUVEdgeProcessingHorizontalDirection(char* input, char* output, int width, int height, double k)	YUVwidth 和 YUVheight 是 输入的 YUV 图像的宽和高, 参考: depth=12, mt=true。 YUV 仅水平方向的边缘处 理, input 是输入文件名, output 是输出文件名。 width 和 height 是输入图 像的宽和高。参考: k=0.7。
<pre>void YUVVieoEdgeProcessing(char* input, char* output, int width, int height, int frame, int max_frame)</pre>	YUV 视频文件边缘处理, input 是输入文件名, output 是输出文件名。 width 和 height 是输入图 像的宽和高, frame 是要处 理的帧序号, max_frame 是 最大帧序号。
<pre>void YUVScale(char* input, char* output, int inputWidth, int</pre>	缩放 yuv420 图像,参考: inputWidth=1280 , inputHeight=720 , outputWidth=128 , outputHeight=72。
<pre>void NoiseTreatment(char* input, char* output, int width, int height, int TWICEwidth, int TWICEheight) void NoiseTreatment(char* input, char* output, int width, int height, int frame, int max_frame)</pre>	YUV 噪声处理。 YUV 噪声处理。

RAW 图像处理

unsigned	char** R	AWRead(char*	RAW 图像读取,返回像素数据。
input, int he	ight, int widt	h)	
void RAW	Write(unsigne	d char**	接收像素数据保存为 RAW 图像。
input, char*	output, int	height, int	
width)			
void ME	SVQ(char*	input, char*	MBVQ 效果, input 是输入文件名,
output, int w	ridth, int heig	ht)	output 是输出文件名。width 和
			height 是输出图像的宽和高。
void RAWtoF	PPM_red(char*	input, char*	RAW 转为 PPM 后提取红色通道,
output, int	width,	int	参考:width=4096,height=3072,
height, DebayerAlgorithm algo)		1go)	algo=NEARESTNEIGHBOUR 或
			LINEAR。支持 RAW12 格式。
			需引入以下枚举:
			enum DebayerAlgorithm {
			NEARESTNEIGHBOUR,
			LINEAR

	};
void RAWtoPPM_green1(char* input, char* output, int width, int	RAW 转为 PPM 后提取绿色 1 通道, 参考: width=4096, height=3072,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举: enum DebayerAlgorithm { NEARESTNEIGHBOUR, LINEAR };
void RAWtoPPM_green2(char*	RAW 转为 PPM 后提取绿色 2 通道,
<pre>input, char* output, int width, int height, DebayerAlgorithm algo)</pre>	参考: width=4096, height=3072, algo=NEARESTNEIGHBOUR 或 LINEAR。
	支持 RAW12 格式。 需引入以下枚举: enum DebayerAlgorithm { NEARESTNEIGHBOUR, LINEAR };
void RAWtoPPM_blue(char* input, char* output, int width, int height, DebayerAlgorithm algo)	RAW 转为 PPM 后提取蓝色通道, 参考: width=4096, height=3072, algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举: enum DebayerAlgorithm { NEARESTNEIGHBOUR, LINEAR };
void RAWtoPPM(char* input, char* output, int width, int height, DebayerAlgorithm algo)	RAW 转为 PPM,参考: width=4096, height=3072 , algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举: enum DebayerAlgorithm { NEARESTNEIGHBOUR, LINEAR };
<pre>void RawPowerTransformation(char* input, char* output, int width, int height, int c, float v)</pre>	幂次变换, input 是输入的 RAW 图像文件名, output 是输出的 RAW 图像文件名, width 是输入图像的

	宽,height 是输入图像的高。默
	认 c=1, v=0.6。支持 RAW 图像。
void RAWAvgFilter(char* input, char*	平均滤波器, input 是输入文件
output, int ROWS, int COLS, int M, float	名,output 是输出文件名。ROWS
mask[3][3])	是图像的行大小,COLS 是图像的
	列大小, M 是滤波相关参数, 如
	M=1; mask 是滤波器模板。支持
	RAW 图像。
	参考模板:
	float $mask[3][3] =$
	{{0.1111, 0.1111, 0.1111},
	((0.1111, 0.1111),
	{0.1111, 0.1111, 0.1111},
	{0.1111, 0.1111, 0.1111}};
void RawImageInversion(char*	图像反相, input 是输入的 RAW 图
input, char* output, int width, int	像文件名,output 是输出的 RAW
height)	图像文件名,width 是输入图像的
	宽,height 是输入图像的高。支
	持 RAW 图像。
void RawHistogramEqualization(char*	直方图均衡化, input 是输入的
input, char* output, int width, int	RAW 图像文件名, output 是输出
height)	的 RAW 图像文件名, width 是输
<i>,</i>	入图像的宽, height 是输入图像
	的高。支持 RAW 图像。
void RAWHistogramEqualization(char*	RAW 直方图均衡化, width 和
input, char* output, int width, int	height 是输入图像的宽和高。
height)	
void RAWMedianFilter(char*	中值滤波, input 是输入文件名,
input, char* output, int ROWS, int	output 是输出文件名。ROWS 是图
COLS, int M, int sequence[9])	像的行,COLS 是图像的列,M 是
oozo, inc m, inc sequence[0],	滤波相关参数,如 M=1。支持 RAW
	図
	int
	sequence $[9] = \{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0$
void RawtoBmp1(char* input, char*	RAW 图像转为 BMP 图像, input 是
output, unsigned long Width, unsigned	输入文件名,output 是输出文件
long Height)	名。Width 和 Height 是输入文件
1010 11010110/	句。"Iden 和 neight 是關人文目 的宽和高。
void RawToBmp(char* input, char*	RAW 图像转为 BMP 图像, input 是
output, int imageWidth, int imageHigth)	输入文件名,output 是输出文件
output, int imagerium, int imagenigth)	名。支持宽和高相等的图像。
	石。又付见州同州寺的图隊。

void RGBtoCMY(string input, string	RGB转CMY,output1是输出的CMY
output1, string output2, int height, int	模型图像名, output2 是输出的
width, int NumberChannels, int a)	CMY 模型图像的单个通道的图像
	名; height 和 width 是输入图像
	input 的高和宽; Number Channels
	是图像的通道数,如通道数为3;
	a=0表示生成 cyan 模型图像,a=1
	表示生成 magenta 模型图像, a=2
	表示生成 yellow 模型图像。支持
	RAW 文件。
void RGBtoHSI(char* input, char*	RGB 模型转为 HIS 模型, input 是
output)	输入文件名, output 是输出文件
output)	名。支持 24 位 BMP 图像。
void SobelOperation1(char*	
`	Sobel 算子。
input, char* output, int width, int	
height)	
void SobelOperation2(char*	Sobel 算子。
input, char* output, int width, int	
height)	
void CyanGray(char* input, char*	青色灰度图像。
output, int width, int height)	
void MagentaGray(char* input, char*	品红灰度图像。
output, int width, int height)	
void YellowGray(char* input, char*	黄色灰度图像。
output, int width, int height)	
void GrayLightness(string	彩色转灰度。
input, string output, int height, int	
width, int NumberChannels)	
void GrayAverage(string input, string	彩色转灰度。
output, int height, int width, int	
NumberChannels)	<i>业</i> 4 大
void GrayLuminosity(string	彩色转灰度。
input, string output, int height, int	
width, int NumberChannels)	
void Transfer(char* input, char*	传递函数。
output, int width, int height)	
void Homography(char* input1,char*	单应。
input2, char* input3, char* output, int	
width, int height, int newwidth, int	
newheight)	
<pre>void MovieEffect(char* input, char*</pre>	电影效果。
output, int width, int height)	
void Dither(string input, string	抖动,参考: method=1,
output, int height, int width, int	bayerMatrixNumber=2 ,
<u> </u>	

NlCl 1 - i t 1 - i t	lOfT0
NumberChannels, int method, int	numberOfTones=2。
bayerMatrixNumber, int numberOfTones)	N -l
void AssimilateChannels(string	抖动,参考: method=1,
input, string output, int height, int	bayerMatrixNumber=2 ,
width, int NumberChannels, int	numberOfTones=2。
method, int bayerMatrixNumber, int	
numberOfTones)	
void FixedThresholdMethod(char*	抖色处理,固定阈值法。
input, char* output, int width, int	
height)	
void RandomThresholdMethod(char*	抖色处理,随机阈值法。
input, char* output, int width, int	
height)	
void DitherMatrixMethod(char*	抖色处理,抖动矩阵法,默认 N=2。
input, char* output, int width, int	
height, int N)	
void NormalizedLogBuffer1(char*	对数变换,规范化对数。
input, char* output, int width, int	
height)	
void NormalizedLogBuffer2(char*	对数变换,规范化对数。
input, char* output, int width, int	
height)	
void TernaryGrayLevel1(char*	三值灰度。
input, char* output, int width, int	
height)	
void TernaryGrayLevel2(char*	三值灰度。
input, char* output, int width, int	
height)	
void BestEdgeMap1(char* input, char*	最佳边贴图。
output, int width, int height)	
void BestEdgeMap2(char* input, char*	最佳边贴图。
output, int width, int height)	
void Skeletonize(char* input, char*	骨架化。
output, int width, int height)	
void GrayLuminosity(string	
input, string output, int height, int	
width, int NumberChannels)	
void	微分图像以获得边缘。
DifferentiateImageSobelFilter(string	
input, string output, int height, int	
width, int NumberChannels)	
void	求反图像以获得边缘映射。
DifferentiateImageSobelFilterAndRGBto	A - M - M - M - M - M - M - M - M - M -
CMY(string input, string output, int	
om totting input, stilling output, int	

woid EdgeDetectionSobelFilter(string input, string output, int height, int width, int threshold) void RemoveSpeckles(string input, string output, int height, int width, int mumberOfflerations, type_morphing morphingOperation) void BoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels, int threshold, int morphingOperation) void DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels, int threshold, string output, int height, int width, int NumberChannels, int threshold, string output, int height, int width, int height input, char* output, int width, int height) void SeparableDiffusion(char* input, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* input, char* output, int width, int height) void Shrink (char* input, char* input, char* output, int width, int height) void Shrink (char* input, char* input, char* output, int width, int height) void Shrink (char* input, char* input, char* output, int width, int height) void Shrink (char* input, char* input, char* input, char* output, int width, int height)		
input, string width, int threshold) void RemoveSpeckles(string input, string output, int height, int threshold) void RemoveSpeckles(string input, string output, int height, int threshold, int numberOflterations, type_morphing morphingOperation) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int height)	height, int width, int NumberChannels)	/
width, int threshold) void RemoveSpeckles(string input, string output, int height, int width, int NumberChannels, int threshold, int numberOfIterations, type_morphing morphingOperation) void DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output, int height, int threshold, string output, int width, int height) void Denoising(char* input, char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int	1	
threshold) void RemoveSpeckles(string input, string output, int height, int width, int numberOflterations, type_morphing morphingOperation) void BilinearTransformation(char* input, char* output, int width, int height) void Denoising(char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* input, char* output, int width, int width, int height)		缘贴图,参考: threshold=50。
woid RemoveSpeckles(string input, string output, int height, int width, int numberOflterations, type_morphing morphingOperation) woid Denoising(char* input, char* output, int width, int height) woid Denoising(char* input, char* output, int width, int height) woid Average (char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void Milmax (char* input, char* output, int width, int height) void Milmax (char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int height)	width, int NumberChannels, int	
input, string output, int height, int width, int numberOflterations, type_morphing morphingOperation) woid	threshold)	
width, int threshold, int numberOfIterations=0 , morphingOperation=THINNING。本函数需添加以下结构体: typedef enum type_morphing	void RemoveSpeckles(string	参考: threshold=-100 ,
threshold, int numberOfIterations, type_morphing morphingOperation) writeHistogramToFile) void Denoising(char* input, char* input, char* output, int width, int height) void Denoising(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void SilinearTransformation(char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int height)	input, string output, int height, int	background=255 ,
numberOfIterations, type_morphing morphingOperation) writeHistogram, bool writeHistogramToFile void SeparableDiffusion(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int wid	width, int NumberChannels, int	numberOfIterations=0 ,
typedef enum type_morphing { SHRINKING = 0x1, THINNING = 0x2, SKELETONIZING = 0x3 } type_morphing; void	threshold, int background, int	morphingOperation=THINNING。
SHRINKING = 0x1, THINNING = 0x2, SKELETONIZING = 0x3 } type_morphing; void DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input], char* input2, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int width, int width, int height)	numberOfIterations, type morphing	本函数需添加以下结构体:
SHRINKING = 0x1, THINNING = 0x2, SKELETONIZING = 0x3 } type_morphing; void DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int width, int height)	morphingOperation)	<pre>typedef enum type morphing {</pre>
THINNING = 0x2, SKELETONIZING = 0x3 } type_morphing; void DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int width, int width, int height)		_
SKELETONIZING = 0x3 } type_morphing; void DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int height)		· ·
void DoubleDifferentiatingGetEdges (string input, string output, int height, int width, int NumberChannels) void GetTernaryMap (string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising (char* input, char* input2, char* output, int width, int height) void Average (char* input, char* output, int width, int height) void MinMax (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void Shrink (char* input, char* output, int width, int height) void BilinearTransformation (char* input, char* output, int width, int height) void BilinearTransformation (char* input, char* output, int width, int height) void BilinearTransformation (char* input, char* output, int width, int height) void BilinearTransformation (char* input, char* output, int width, int width, int height) void BilinearTransformation (char* input, char* output, int width, int width, int height) void BilinearTransformation (char* input, char* output, int width, in		· ·
woid DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int height)		
DoubleDifferentiatingGetEdges(string input, string output, int height, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int height)	void	
input, string output, int width, int width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		TO THE PARTY PARTY OF THE PARTY
width, int NumberChannels) void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* inputl, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		
input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		创建三元边贴图 参老.
width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int	1 \	
threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int width, int		
output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int width, int height)		wiltenistogramiorile-laise.
writeHistogramToFile) void SeparableDiffusion(char* 可分离扩散。 input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		
void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* 去除噪声。 input2, char* output, int width, int height) void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		
input, char* output, int width, int height) void Denoising(char* input1, char* 去除噪声。 input2, char* output, int width, int height) void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int		可以该护斯
height) void Denoising(char* input1, char* 去除噪声。 input2, char* output, int width, int height) void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int	i '	円分割 取。
void Denoising(char* input1, char* 去除噪声。 input2, char* output, int width, int height) void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		
input2, char* output, int width, int height) void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* 平均化。 output, int width, int height) void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* input, char* output, char* output, int width, int width, int		+ 公県 幸
height) void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* output, int width, int height) void MinMax(char* input, char* output, int width, int height) void Shrink(char* input, char* output, int width, int height) void BilinearTransformation(char* input, char* output, int width, int width, int width, int		
void Luminosity(char* input, char* 亮度调整。 output, int width, int height) void Average(char* input, char* 平均化。 output, int width, int height) void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int		
output, int width, int height) void Average(char* input, char* 平均化。 output, int width, int height) void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int		
void Average(char* input, char* 平均化。 output, int width, int height) void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int	1	
output, int width, int height) void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int		
void MinMax(char* input, char* 最小与最大。 output, int width, int height) void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int		平均化。
output, int width, int height) void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int	output, int width, int height)	
void Shrink(char* input, char* 收缩。 output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int	void MinMax(char* input, char*	最小与最大。
output, int width, int height) void BilinearTransformation(char* 双线性变换。 input, char* output, int width, int	output, int width, int height)	
void BilinearTransformation(char* 双线性变换。 input,char* output,int width,int	void Shrink(char* input, char*	收缩。
input, char* output, int width, int	output, int width, int height)	
	void BilinearTransformation(char*	双线性变换。
height, int newwidth, int newheight)	input, char* output, int width, int	
	height, int newwidth, int newheight)	
void BilinearInterpolation(string 双线性插值,参考:	void BilinearInterpolation(string	双线性插值,参考:
input, string output, int height, int targetHeight=200 ,	input, string output, int height, int	targetHeight=200 ,

width, int NumberChannels, int	targetWidth=200。
	targetwidth-200°
targetHeight, int targetWidth)	
void DitherMatrixMethod(char*	四级抖动,默认 N=2。
input, char* output, int width, int	
height, int N)	国 版 上 <i>山</i> コ
int ObjectsInImages(string	图像中的对象,参考:
input, string output, int height, int	threshold=128
width, int NumberChannels, int	starSize=1.5833, a=1 表示确保
threshold, float starSize, int a)	前背景为黑色。支持 RAW 文件。
void Dewarped1(char* input, char*	脱蜡。a 是在输出图像中检查半径
output, int width, int height, int	是否<=a, 然后再进行扭曲,参考:
Offset, double a, double b)	Offset=256, a=256.5, b=0.5。
<pre>void Dewarped2(char* input, char*</pre>	脱蜡。a 是在输出图像中检查半径
output, int width, int height, int	是否<=a,然后再进行扭曲,参考:
Offset, double a, double b, double	Offset=256, a=256.5, b=0.5。
<pre>coeffx[12], double coeffy[12])</pre>	脱蜡规范:
	double coeffx[12] =
	{ 1.00056776e+00, -
	5.68880703e-04, -
	1. 13998357e-03,
	1.00056888e+00,-
	5. 65549579e-04, -1. 13554790e- 03,
	9. 99434446e-
	01 , 5. 66658513e-04 ,
	1. 13110351e-03 ,
	9. 99433341e-
	01 , 5. 67767429e-04 ,
	1. 13553921e-03 };
	1.133333216 03),
	double coeffy[12] = {-
	5. 67763072e=04,
	1. 00056888e+00, 1. 13998357e-
	ŕ
	5 69990702
	5. 68880703e-
	04, 9.99434450e-01, -
	1. 13554790e-03,
	5. 65553919e-
	04, 9.99433341e-01, -
	1. 13110351e-03,
	-5. 66658513e-
	04, 1.00056777e+00,
	1.13553921e-03};

	사고비 사 살다. 뛰이나 표 그 그 그 그 그
void TextureSegmentation1(char*	纹理分割,默认 K=6, №100。
input, char* output, int width, int	
height, int K, int N)	
void TextureSegmentation2(char*	纹理分割,默认 K=6, N=100。
input, char* output, int width, int	
height, int K, int N)	
void TextureClassification(vector	纹理分类,a是要分类的图像的数
<pre><string> filename, char* output, int</string></pre>	量,如 filename 里有 3 个图像名
width, int height, int K, int N, int a)	称,则 a=3; output 是分类结果
	文件,格式为 txt 的文本文件;
	默认 K=4, N=1000。
void ErrorDiffusion1(char*	误差扩散。
input, char* output, int width, int	
height)	
void ErrorDiffusion2(char*	误差扩散。
input, char* output, int width, int	
height)	
void ErrorDiffusion3(char*	误差扩散。
input, char* output, int width, int	
height)	
void ErrorDiffusion(string	误差扩散,参考: method=1,
input, string output, int height, int	kernelSize=3 ,
width, int NumberChannels, int	numberOfTones=2 ,
method, int kernelSize, int	useFilter=false。支持 RAW 文
numberOfTones, bool useFilter)	件。
void Thin(char* input, char*	图像细化。
output, int width, int height)	
void	形态滤波。
BinaryMorphologicalFilteringComplete(参考: threshold=100 ,
string input, string output, int	numberOfIterations=0 ,
height, int width, int	morphingOperation=(type_morp
NumberChannels, int threshold, int	hing)3。支持 RAW 文件。
numberOfIterations, type_morphing	
morphingOperation)	
void OilPainting(char* input, char*	油画效果,默认 N=2。
output, int width, int height, int N)	
void OilPainting1(char* input, char*	油画效果,默认 N=2。
output, int width, int height, int N)	
void 0ilPaintingEffect1(string	油画效果,参考: colorBits=3,
input, string output, int height, int	bitsOK=true, kernelSize=9,
width, int NumberChannels, int	kernelSizeOK=true。支持 RAW 文
colorBits, bool bitsOK, int	件。
kernelSize, bool kernelSizeOK)	
void 0ilPaintingEffect2(string	油画效果,参考: colorBits=3,
	油画效果,参考: colorBits=3,

input, string output, int height, int width, int colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) void 0ilPaintingEffect3(string input, string output, int height, int width, int colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) width, int NumberChannels, int colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
colorBits, bool kernelSizeOK) void OilPaintingEffect3(string 油画效果,参考: colorBits=3, input, string output, int height, int width, int colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
kernelSize, bool kernelSizeOK) void 0ilPaintingEffect3(string 油画效果,参考: colorBits=3, input, string output, int height, int width, int NumberChannels, int colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
void 0ilPaintingEffect3(string 油画效果,参考: colorBits=3, input, string output, int height, int width, int colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
input, string output, int height, int width, int NumberChannels, int colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
width, int colorBits, bool bitsOK, int kernelSizeOK=true。支持RAW文 件。 kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
colorBits, bool bitsOK, int kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
kernelSize, bool kernelSizeOK) void AverageFiltering(char* 3*3 平均值滤波。
void AverageFiltering(char* 3*3 平均值滤波。
inputfile, char* outputfile, int
width, int height)
void GeometricMeanFiltering(char* 3*3 几何均值滤波。
inputfile, char* outputfile, int
width, int height)
void MedianFiltering(char* 中值滤波。
inputfile, char* outputfile, int
width, int height)
void FFT(char* input, char* output, int FFT 函数。
width, int height)
void LowPassOrHighPassFiltering(char* 低通或高通滤波。LOW_PASS=1 为
input, char* output, int width, int 低通滤波, 否则为高通滤波,
height, int LOW_PASS, int DEGREE)
DEGREE=0.
void IFFT (char* input, char* IFFT 函数。LOW_PASS=1 为低通滤
output, int width, int height, int 波, 否则为高通滤波, DEGREE 为
LOW_PASS, int DEGREE) 滤波程度,如 DEGREE=0。
void BMPtoRAW(char* inputfile, char* BMP 图像转 RAW 图像。支持 24 为
outputfile) BMP 图像。
void BMPtoRAW1 (char* input, char* BMP 图像转 RAW 图像。支持 24 为
output) BMP 图像。

BMP 图像处理

unsigned char**	读取 8 位 BMP 图像的像素。
BMPRead8(char* input)	
void GenerateImage8(char*	生成 8 位 BMP 图像, output 是生成的图像
output, unsigned char**	文件名,color 是像素数据。
color)	
BMPMat** BMPRead(char*	读取 24 位和 32 位 BMP 图像的像素。
input)	需要引入以下结构体:
	typedef struct {
	unsigned char B; //24位和32位BMP
	图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP

B像的绿色通道分量		
図像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的 Alpha 通道 } BMPMat; unsigned int BMPHeight(char* iput) unsigned int BMPWidth(char* iput) unsigned int BMPWidth(char* iput) unsigned int BMPWidth(char* iput) unsigned int BMPWidth(char* iput) woid GenerateImage(char* output, BMPMat** color, unsigned short type)		图像的绿色通道分量
unsigned char A; //仅限 32 位 BMP 图像的 Alpha 通道 BMPMat; input) unsigned int BMPHeight(char* input) void GenerateImage(char* output, BMPMat** color, unsigned short type) ### With a provided and a provid		unsigned char R; //24 位和 32 位 BMP
## Alpha 通道 BMPMat; unsigned int BMPWidth(char* 读取 BMP 图像的离度。 input) void GenerateImage(char* output, BMPWat** color, unsigned short type) **BMPMat*** color, unsigned short type) **BMPMat*** **BMPMat*** **Color = (BMPMat**) malloc(sizeof(BMPMat*)*1280); for (unsigned int i = 0; i < 1280; i++) {		图像的红色通道分量
BMPMat;		unsigned char A; //仅限 32位BMP图
wisigned int BMPHeight (char* input) unsigned int BMPWidth (char* input) wid GenerateImage (char* output, BMPMat** color, unsigned short type) ### ### ### ### ### ### ### ### ### #		像的 Alpha 通道
input) unsigned int BMPWidth(char* input) void GenerateImage(char* color, unsigned short type) BMPMat*** color, unsigned short type) BMPMat*** color [BMPMat***] (BMPMat**) malloc(sizeof(BMPMat*)*1280); for (unsigned int i = 0; i < 1280; i++) {		}BMPMat;
input) unsigned int BMPWidth(char* input) void GenerateImage(char* color, unsigned short type) BMPMat*** color, unsigned short type) BMPMat*** color [BMPMat**] (BMPMat**) malloc(sizeof(BMPMat*)*1280); for (unsigned int i = 0; i < 1280; i++) {	unsigned int BMPHeight(char*	读取 BMP 图像的高度。
wisigned int BMPWidth (char* jxx BMP 图像的宽度 input	_	
input) void GenerateImage(char* output, BMPMat** color, unsigned short type) BMPMat** color = (BMPMat**) malloc(sizeof(BMPMat*)*1280);		│ │
void GenerateImage(char* output, BMPMat** 生成 24 位和 32 位 BMP 图像。type 等于图像的位数,如 type=24。参考用例:	_	以"人"的"国际门"是人
output, BMPMat** color, unsigned short type) 参考用例: BMPMat** color = (BMPMat**) malloc (sizeof (BMPMat*)*1280); for (unsigned int i = 0; i < 1280; i++) { color[i] = (BMPMat*) malloc (sizeof (BMPMat)*2450); } for (unsigned int i = 0; i < 1280; i++) { color[i][j].B = 0; color[i][j].G = 0; color[i][j].R = 255; } void HistogramEqualization5 (char* input, char* output) void Resize (char* input, char* output, int Height, int Width) HistogramEqualization5 (char* input, char* output, int Height, int Width) double MeanBrightness (char* input) int IsBitMap (FILE *fp) int getWidth (FILE *fp) int getWidth (FILE *fp) int getWidth (FILE *fp) int getHeight (FILE *fp) was part of the mode of the mall of the mode of the mod		上式 94 位和 29 位 DMD 图像 +wp。 竿王图
color, unsigned short type)		
BMPMat** color = (BMPMat**) malloc (sizeof (BMPMat*)*1280); for (unsigned int i = 0; i < 1280; i++) {	_	
(BMPMat**) malloc(sizeof(BMPMat*)*1280); for (unsigned int i = 0; i < 1280; i++) {	color, unsigned short type)	
Color[i]		
for (unsigned int i = 0; i < 1280; i++) {		
i++));
Color[i]		for (unsigned int $i = 0$; $i < 1280$;
color[i]		i++)
(BMPMat*)malloc(sizeof(BMPMat)*2450); } for (unsigned int i = 0; i < 1280; i++) { for (unsigned int j = 0; j < 2450; j++) { color[i][j].B = 0; color[i][j].R = 255; } void HistogramEqualization5(char* input, char* output) void Resize(char* input, char* output, int Height, int Width) Height, int Width) double MeanBrightness(char* input) vid Resize(char* input, char* output, int Height, int Width) Height, int Width) double MeanBrightness(char* input) int IsBitMap(FILE *fp) int getWidth(FILE *fp) int getHeight(FILE *fp) was alloc (sizeof(BMPMat)*2450); if the color in the		{
Solution File For (unsigned int i = 0; i < 1280; i++) { for (unsigned int j = 0; j < 2450; j++) { color[i][j].B = 0; color[i][j].G = 0; color[i][j].R = 255; Solution Equalization Equ		color[i] =
for (unsigned int i = 0; i < 1280; i++) {		(BMPMat*)malloc(sizeof(BMPMat)*2450);
i++) { for (unsigned int j = 0; j < 2450; j++) { color[i][j].B =0; color[i][j].G =0; color[i][j].R =255; } } void HistogramEqualization5(char* input, char* output) void Resize(char* input, char* output, int Height, int Width) double MeanBrightness(char* input) double MeanBrightness(char* input) double MeanBrightness(char* input) int IsBitMap(FILE *fp) jint getWidth(FILE *fp) jint getWidth(FILE *fp) int getHeight (FILE *fp) unsigned short getBit(FILE * 获得每个像素的位数。		}
i++) { for (unsigned int j = 0; j < 2450; j++) { color[i][j].B =0; color[i][j].G =0; color[i][j].R =255; } } void HistogramEqualization5(char* input, char* output) void Resize(char* input, char* output, int Height, int Width) double MeanBrightness(char* input) double MeanBrightness(char* input) double MeanBrightness(char* input) int IsBitMap(FILE *fp) jint getWidth(FILE *fp) jint getWidth(FILE *fp) int getHeight (FILE *fp) unsigned short getBit(FILE * 获得每个像素的位数。		for (unsigned int $i = 0$: $i < 1280$:
for (unsigned int j = 0; j < 2450; j++) {		
2450; j++) {		
color[i][j].B =0; color[i][j].G =0; color[i][j].R =255; } void HistogramEqualization5(char* input, char* output) void Resize(char* input, char* output, int Height, int Width) double MeanBrightness(char* input) double MeanBrightness(char* input) int IsBitMap(FILE *fp) int getWidth(FILE *fp) int getHeight(FILE *fp) wright color[i][j].B =0; colo		
color[i][j].G =0; color[i][j].R =255; } void HistogramEqualization5(char* input, char* output) void Resize(char* input, char* output, int Height, int Width) double MeanBrightness(char* input) vidtle MeanBrightness(char* input) double MeanBrightness(char* input) int IsBitMap(FILE *fp) int getWidth(FILE *fp) int getHeight(FILE *fp) wright color[i][j].G =0; color[i		
color[i][j].R =255; } void HistogramEqualization5(char*		_
void HistogramEqualization5(char* input, char* output) void Resize(char* input, char* output) void Resize(char* input, char* output, int Height, int Width) double MeanBrightness(char* input) int IsBitMap(FILE *fp) int getWidth(FILE *fp) unsigned short getBit(FILE 获得每个像素的位数。		_
HistogramEqualization5(char* input, char* output) void Resize(char* 图片缩放,支持8位和16位BMP。input是 输入文件名,output是输出文件名。Height height, int Width) double MeanBrightness(char* input) int IsBitMap(FILE*fp) int getWidth(FILE*fp) int getHeight(FILE*fp) unsigned short getBit(FILE 获得每个像素的位数。		color[1][j].R =255;
HistogramEqualization5(char* input, char* output) void Resize(char* 图片缩放,支持8位和16位BMP。input是 输入文件名,output是输出文件名。Height height, int Width) double MeanBrightness(char* input) int IsBitMap(FILE*fp) int getWidth(FILE*fp) int getHeight(FILE*fp) unsigned short getBit(FILE 获得每个像素的位数。		}
HistogramEqualization5(char* input, char* output) void Resize(char* 图片缩放,支持8位和16位BMP。input是 输入文件名,output是输出文件名。Height height, int Width) double MeanBrightness(char* input) int IsBitMap(FILE*fp) int getWidth(FILE*fp) int getHeight(FILE*fp) unsigned short getBit(FILE 获得每个像素的位数。		}
input, char* output) void Resize(char* 图片缩放,支持 8 位和 16 位 BMP。input 是	void	-
voidResize(char* input, char* output, int Height, int Width)图片缩放,支持8位和16位BMP。input是输出文件名。Height和Vidth是输出图像的高和宽。double MeanBrightness(char* 	HistogramEqualization5(char*	是输入文件名,output 是输出文件名。
input, char* output, int Height, int Width	input, char* output)	
input, char* output, int Height, int Width	void Resize(char*	图片缩放,支持8位和16位BMP。input是
Height, int Width)和 Width 是输出图像的高和宽。double MeanBrightness (char* input)求图像的平均亮度,支持8位和16位BMP。int IsBitMap(FILE*fp)判断是否是位图。int getWidth(FILE*fp)获得图片的宽度。int getHeight(FILE*fp)获得图片的高度。unsigned short getBit(FILE获得每个像素的位数。	·	
double MeanBrightness(char* 求图像的平均亮度,支持8位和16位BMP。input)		- · · · · · · · · · · · · · · · · · · ·
input)input 是输入文件名。int IsBitMap(FILE *fp)判断是否是位图。int getWidth(FILE *fp)获得图片的宽度。int getHeight(FILE *fp)获得图片的高度。unsigned short getBit(FILE 获得每个像素的位数。		
int IsBitMap(FILE *fp)判断是否是位图。int getWidth(FILE *fp)获得图片的宽度。int getHeight(FILE *fp)获得图片的高度。unsigned short getBit(FILE 获得每个像素的位数。	· ·	
int getWidth(FILE *fp) 获得图片的宽度。 int getHeight(FILE *fp) 获得图片的高度。 unsigned short getBit(FILE 获得每个像素的位数。		
int getHeight(FILE *fp) 获得图片的高度。 unsigned short getBit(FILE 获得每个像素的位数。		
unsigned short getBit(FILE 获得每个像素的位数。		
	,	
*fp)		
	*fp)	

,	
unsigned int getOffSet(FILE	获得数据的起始位置。
*fp)	
void BMPtoYUV(char*	BMP 图像转为 YUV 图像, input 是输入文件
input, char* output, char	名,output 是输出文件名。yuvmode 是 YUV
yuvmode)	文件的 3 个模式选项,yuvmode 的值可为
	'0'、'2'、'4',分别为 420,422,444
void BMPtoYUV420I(char*	BMP 图像转为 YUV420 图像, input 是输入文
input, char* output)	件名,output 是输出文件名。
void BMPtoYUV420II(char*	BMP 图像转为 YUV420 图像, input 是输入文
input, char* output)	件名,output 是输出文件名。
void DCMtoBMP(string	DCM 图像转 BMP 图像。input 是输入文件名,
input, char* output)	output 是输出文件名。
void Ins1977 (char*	Ins1977 滤镜, input 是输入文件名, output
input, char* output, int	是输出文件名。参考: ratio=100。
ratio)	
void LOMO(char* input, char*	LOMO 滤镜,DarkAngleInput 是暗角模板图
DarkAngleInput, char*	像名,参考: ratio=100。
output, int ratio)	
void PNGGray(char*	图像灰度化,input 是输入文件名,output
input, char* output)	是输出文件名。
void PNGSpotlight(char*	聚光灯效果, input 是输入文件名, output
input, char* output, int	是输出文件名。焦点坐标
centerX, int centerY, double	(centerX,centerY) , 如 :
a, double b, double c, double	centerX=400, centerY=180; a, b, c, d, e
d, double e)	是相关参数,默认 a=100,b=100,c=160,
	d=80, e=0.5.
void PNGIllinify(char*	幻化效果, input 是输入文件名, output 是
input, char* output)	输出文件名。
void PNGWaterMark(char*	图像加水印, input1 和 input2 的尺寸必须
input1, char* input2, char*	相同。
output)	
void Short(char* input, char*	矮化特效。a=1,b=128,c=2,d=0.5,
output, int a, int b, int	depth=24。支持 24 位 BMP 图像。
c, double d, int depth)	
void Rise(char* input, char*	增高特效。a=1, b=128, c=0.5, d=2,
output, int a, int b, double	depth=24。支持 24 位 BMP 图像。
c, int d, int depth)	
void Short1(char*	矮小化特效。a=1, b=128, c=0.5, d=0.5,
input, char* output, int a, int	depth=24。支持 24 位 BMP 图像。
b, double c, double d, int	
depth)	
void Handstand(char*	倒立特效。a=1, b=128, c=0.5, depth=24。
input, char* output, int a, int	支持 24 位 BMP 图像。
b, double c, int depth)	

void Fat(char* input, char*	肥胖特效。a=1, b=128, c=0.5, depth=24。
output, int a, int b, double	支持 24 位 BMP 图像。
c, int depth)	
void HighFoot(char*	高脚特效。a=1, b=128, c=2, d=0.5,
input, char* output, int a, int	depth=24。支持 24 位 BMP 图像。
b, int c, double d, int depth)	30F 211 212 2111 2111
void CurvedCurve(char*	弧度弯曲特效。a=1,b=128,c=4,d=2,e=0.5,
,	
input, char* output, int a, int	depth=24。支持 24 位 BMP 图像。
b, int c, int d, double e, int	
depth)	
void Thin(char* input, char*	细化特效。a=1, b=128, c=0.5, d=0.5,
output, int a, int b, double	depth=24。支持 24 位 BMP 图像。
c, double d, int depth)	
void Winding(char*	弯曲特效。1im=20, a=1, b=128, c=4, d=5,
input, char* output, int	e=0.5,depth=24。支持 24 位 BMP 图像。
lim, int a, int b, int c, int	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
d, double e, int depth)	
void CrossDenoising (unsigned	十字法剔除孤立像素点。
char** input, unsigned char**	需引入以下结构体和声明:
output, double a)	typedef struct {
	unsigned char B; //24 位和 32 位 BMP
	图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP
	图像的绿色通道分量
	unsigned char R; //24 位和 32 位 BMP
	图像的红色通道分量
	unsigned char A; //仅限 32位 BMP 图
	像的 Alpha 通道
	}BMPMat;
	typedef struct {
	double B;
	double G;
	double R;
	·
	double A;
	}BMPMatdouble;
	void Conversion8(unsigned char**
	input, double** output);
	void Conversion8(double**
	input, unsigned char** output);
	void Conversion24(BMPMat**
	input,BMPMatdouble** output);
	void Conversion24(BMPMatdouble**
	Dill ild backeto.

	input, BMPMat** output):
<pre>void CrossDenoising(BMPMat** input,BMPMat** output,double a)</pre>	input, BMPMat** output); 十字法剔除孤立像素点。 需引入以下结构体和声明: typedef struct { unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量 unsigned char G; //24 位和 32 位 BMP 图像的绿色通道分量 unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图 像的 Alpha 通道 } BMPMat;
	<pre>typedef struct { double B; double G; double R; double A; } BMPMatdouble; void Conversion8(unsigned char** input, double** output); void Conversion8(double** input, unsigned char** output); void Conversion24(BMPMat** input, BMPMatdouble** output); void Conversion24(BMPMatdouble**</pre>
void CrossConnectionDenoising(uns igned char** input, unsigned char** output, double a)	input, BMPMat** output); 交叉法剔除孤立像素点。 需引入以下结构体和声明: typedef struct { unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量 unsigned char G; //24 位和 32 位 BMP 图像的绿色通道分量 unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图 像的 Alpha 通道 } BMPMat;
	<pre>typedef struct { double B;</pre>

	double G;
	double R;
	double A;
	BMPMatdouble;
) DMF Matdouble,
	world Convergion Quantity about
	void Conversion8(unsigned char**
	input, double** output);
	void Conversion8(double**
	input, unsigned char** output);
	void Conversion24(BMPMat**
	input,BMPMatdouble** output);
	void Conversion24(BMPMatdouble**
	input,BMPMat** output);
void	交叉法剔除孤立像素点。
CrossConnectionDenoising(BMP	需引入以下结构体和声明:
Mat** input, BMPMat**	typedef struct {
output, double a)	unsigned char B; //24位和32位BMP
	图像的蓝色通道分量
	unsigned char G; //24位和 32位BMP
	图像的绿色通道分量
	unsigned char R; //24 位和 32 位 BMP
	图像的红色通道分量
	unsigned char A; //仅限 32 位 BMP 图
	像的 Alpha 通道
	BMPMat;
	J Dill Mate,
	typedef struct {
	double B;
	double G;
	double R;
	,
	double A;
	}BMPMatdouble;
	11 C
	void Conversion8(unsigned char**
	input, double** output);
	void Conversion8(double**
	input, unsigned char** output);
	void Conversion24(BMPMat**
	input,BMPMatdouble** output);
	void Conversion24(BMPMatdouble**
	input,BMPMat** output);
void	矩阵法剔除孤立像素点。
MatrixDenoising(unsigned	需引入以下结构体和声明:
char** input, unsigned char**	typedef struct {

output, double a)	unsigned char B; //24 位和 32 位 BMP
	图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP
	图像的绿色通道分量
	unsigned char R; //24 位和 32 位 BMP
	图像的红色通道分量
	unsigned char A; //仅限 32 位 BMP 图
	像的 Alpha 通道
	}BMPMat;
	typedef struct {
	double B;
	double G;
	double R;
	double A;
	}BMPMatdouble;
	void Conversion8(unsigned char**
	input, double** output);
	void Conversion8(double**
	input, unsigned char** output);
	void Conversion24 (BMPMat**
	input, BMPMatdouble** output);
	input, BMPMat** output);
void	矩阵法剔除孤立像素点。
MatrixDenoising(BMPMat**	需引入以下结构体和声明:
input, BMPMat** output, double	typedef struct {
a)	unsigned char B; //24 位和 32 位 BMP
	图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP
	图像的绿色通道分量
	unsigned char R; //24位和32位BMP
	图像的红色通道分量
	unsigned char A; //仅限 32位 BMP 图
	像的 Alpha 通道
	BMPMat;
	typedef struct {
	double B;
	double G;
	-
	double R;
	double A;
	}BMPMatdouble;

<pre>void</pre>	void Conversion8(unsigned char** input, double** output); void Conversion8(double** input, unsigned char** output); void Conversion24(BMPMat** input, BMPMatdouble** output); void Conversion24(BMPMatdouble** input, BMPMat** output); 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 ImageFusion4(char*	多聚焦图像的融合,支持 8 位 BMP 图像。
input1, char* input2, char*	タ永点国家印版日,又17 6 位 DMI 国家。 block_height=8 , block_width=8 ,
output, int block_height, int	threshold=1.75.
block_width, double	
threshold)	
void ImageFusion5(char*	图像融合。参考: a=3, b1=4, DX1=-68, DY1=-
input1, char* input2, char*	99, EPS=1, input1="图像融合 1.jpg",
MaskImage, char* output, int	input2="图像融合 2.jpg", MaskImage="掩
dx[], int dy[], int a, double	膜.png", output="output.jpg"。
b1, int DX1, int DY1, double	int $dx[] = \{0, 0, -1, 1\};$
EPS)	int $dy[] = \{-1, 1, 0, 0\};$
void Dark(char* input, char*	暗调滤镜,参考: ratio=100。
output, int ratio)	
void WaveFilter(char*	
input, char* output, int	浪扭曲度)。a=0 时生成 BMP 图像,a=1 时生
degree, int a)	成 JPG 图像,a=2 时生成 PNG 图像,a=3 时
	生成 TGA 图像,参考: degree=10。
void PinchFilter(char*	挤压形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int a)	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
	a=3 时生成 TGA 图像。
void PinchFilter(char*	挤压形变特效滤镜,a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像,a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
	坐标,cenY 是形变中心点Y 坐标。
void SpherizeFilter(char*	球面形变特效滤镜,a=0 时生成 BMP 图像,
input, char* output, int a)	a=1 时生成 JPG 图像,a=2 时生成 PNG 图像, a=3 时生成 TGA 图像。
void SpherizeFilter(char*	球面形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
cena, int ceni, int a)	α-ση工从 τυλ 国隊, CellA 定形文中心从 λ

	坐标,cenY 是形变中心点 Y 坐标。
void SwirlFilter(char*	旋转形变特效滤镜,a=0时生成BMP图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
ratio, int a)	a=3 时生成 TGA 图像,ratio=3。
void SwirlFilter(char*	旋转形变特效滤镜,a=0时生成BMP图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int ratio, int	a=3 时生成 TGA 图像, ratio=3, cenX 是形
a)	变中心点 X 坐标, cenY 是形变中心点 Y 坐
	标。
void ClosedOperation(char*	闭运算, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 4 位 BMP 图像。
void AdjustPixel(char*	调整像素值, input 是输入文件名, output
input, char* output, int a)	是输出文件名。a 是用于设置图像像素的相
	关参数,如 a=3。支持 24 位 BMP 图像。
void	部分颜色保留滤镜,参考: ratio=60。
PartialColorRetention(char*	
input, char* output, int	
ratio)	
void	生成图像的灰度图,支持 8 位 BMP 图像。
GrayImageConversion8(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。
void Gray(char* input, char*	灰度图转换,支持24位BMP图像。input是
output)	输入文件名,output 是输出文件名。
void	彩色图转灰度图, input 是要处理的彩色图
GrayImageConversion(char*	像, output 是处理后生成的灰度图名称。支
input, char* output)	持 24 位 BMP 图像。
void	二值图像垂直镜像, input 是输入图像的像
BinaryImageVerticalMirror(un	素数据, output 是输出图像的像素数据, w
signed char *input, unsigned	是输入图像的宽, h 是输入图像的高。
char *output, unsigned int	
w,unsigned int h)	
void	灰度图像垂直镜像, input 是输入图像的像
GrayImageVerticalMirror(unsi	素数据, output 是输出图像的像素数据, w
gned char *input,unsigned	是输入图像的宽,h 是输入图像的高。
char *output, unsigned int	
w, unsigned int h)	
void	彩色图像垂直镜像, input 是输入图像的像
ColorImageVerticalMirror(uns	素数据, output 是输出图像的像素数据, w
igned char *input,unsigned	是输入图像的宽,h 是输入图像的高。
char *output, unsigned int	
w, unsigned int h)	
void OTSU(char* input, char*	大津算法, input 是输入文件名, output 是
output, int BeforeThreshold)	输出文件名。BeforeThreshold 是初始阈
	值,如 BeforeThreshold=10。支持 8 位 BMP

	图像。
void LowerBrightness(char*	调低亮度, input 是输入文件名, output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100, b=0。
void HightBrightness(char*	调高亮度, input 是输入文件名, output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100, b=0。
void	迭代阈值选择, input 是输入文件名,
IterativeThresholdSelection(output 是输出文件名。支持 8 位 BMP 图像。
char* input, char* output)	
void DitheringMethod(char*	抖动法, input 是输入文件名, output 是输
input, char* output)	出文件名。支持8位BMP图像。
void LogTransformation(char*	对数变换, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。constant
constant)	是相关参数,如 constant=15。
void	对数变换, input 是输入文件名, output 是
LogarithmicTransformation(ch	输出文件名。支持 BMP 图像。
ar* input, char* output)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization(char*	output 是输出文件名。支持 BMP 图像。
input, char* output)	
void Binarization(char*	二值化, input 是输入文件名, output 是输
input, char* output, int	出文件名。支持 24 位 BMP 图像。threshold
threshold)	是阈值,如: threshold=128。
void Expansion(char*	二 值 图 像 膨 胀 , 参 考 :
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。
void Corrosion(char*	二值图像腐蚀,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。
void OpenOperation(char*	二值图像开运算,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。
void ClosedOperation(char*	二值图像闭运算,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。
void	二值图像开运算提取轮廓,参考:
OpenOperationToExtractContou	$mask[9] = \{0, 255, 0, 255, 255, 255, 0, 255, 0\}$
r(char* input, char*	, c=128。
output, unsigned char	
mask[9], int c)	
void	二值图像膨胀运算提取轮廓,参考:
ExpansionOperationToContourE	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
xtraction(char* input, char*	, c=128。

	1
output, unsigned char	
mask[9], int c)	
void	二值图像腐蚀运算提取轮廓,参考:
CorrosionCalculationToContou	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
rExtraction(char*	, c=128。
input, char* output, unsigned	
char mask[9], int c)	
void Glaw(char* input, char*	发光滤镜,参考: ratio=100。
output, int ratio)	
void LowPassFilter(char*	低通滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 BMP 图像。
void HighPassFilter(char*	高通滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 BMP 图像。
void Thinning(char*	图像细化, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 BMP 图像。
void ThinningLine(char*	图像细化且线条化, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 BMP 图像。
void Corrosion(char*	腐蚀, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 4 位 BMP 图像。
void Corrosion1(char*	腐蚀, input 是输入文件名, output 是输出
input, char* output, int	文件名。支持 24 位 BMP 图像。TempBuf 是
*TempBuf, int TempH, int	腐蚀模板,TempH 和 TempW 分别是 TempBuf
TempW)	的高和宽,如 TempH=4,TempW=4,则有
	TempBuf[4][4].
void Expand(char*	膨胀, input 是输入文件名, output 是输出
input, char* output, int	文件名。支持 24 位 BMP 图像。TempBuf 是
*TempBuf, int TempH, int	膨胀模板,TempH 和 TempW 分别是 TempBuf
TempW)	的高和宽,如 TempH=4,TempW=4,则有
	TempBuf[4][4].
unsigned char**	线性存储的灰阶图像像素转化为二维。
create2DImg(unsigned char*	
input, int w, int h)	
unsigned char	图像指定区域取最大值(判断是否超出边
getMaxPixelWhole(unsigned	界)。
char **input, int x, int y, int	
w, int h, int *Kernal, int	
kernalW, int halfKernalW)	
unsigned char	图像指定区域取最大值(不判断是否超出边
getMaxPixelCenter(unsigned	界)。
char **input, int x, int y, int	
*Kernal, int kernalW, int	
halfKernalW)	
unsigned char**	图像膨胀。
imgDilate(unsigned char	
	

*input, int w, int h, int	
*Kernal, int kernalW, int	
halfKernalW)	
unsigned char	图像指定区域取最小值(判断是否超出边
getMinPixelWhole(unsigned	界)。
char **input, int x, int y, int)1 /0
w, int h, int *Kernal, int	
kernalW, int halfKernalW)	
unsigned char	图像指定区域取最小值(不判断是否超出边
getMinPixelCenter(unsigned	界)。
char **input, int x, int y, int	
*Kernal, int kernalW, int	
halfKernalW)	
unsigned char**	图像腐蚀。
imgErode(unsigned char	
*input, int w, int h, int	
*Kernal, int kernalW, int	
halfKernalW)	
void Corrosion (unsigned char	二值腐蚀。
*input, unsigned char	
cols, int mat[5][5])	一は中間
void Expansion(unsigned char	二值膨胀。
*input, unsigned char	
*output, int rows, int	
cols, int mat[5][5])	
void BoxBlurAdvanced(string	高级方框模糊,参考: radius=5。支持 PNG
input, string output, int	文件。
radius)	
void	高斯滤波,支持 PNG 文件。
GaussianBlurFilter(char*	
input, char* output)	
void GaussianFiltering(char*	高斯滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void	拉普拉斯增强, input 是输入文件名,
LaplaceEnhancement(char*	output 是输出文件名。支持 24 位 BMP 图
input, char* output)	像。
void Residual(char*	求残差, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void SunlightFilter(char*	光照特效滤镜, intensity 是光照强度, 如:
input, char* output, int	intensity=255; radius 是光照范围,如:
intensity, int radius, int	radius=600; x 和 y 是光照的位置,如:
x, int y)	x=100, y=60 _o
void Compress(char*	压缩,支持多种文件。input 是要压缩的文

input, char* output)	件名,output 是压缩后的文件名。
void Decompression(char*	解压缩,支持多种文件。input 是要解压缩
input, char* output)	的文件名,output 是解压缩后的文件名。
void BlackWhite(char*	黑白化,参考: threshold1、threshold2和
input, char* output, int	threshold3 都等于 128, color1=255,
width, int height, unsigned	color2=0。支持 24 位 BMP 图像。
char thresholdl, unsigned	
char threshold2, unsigned	
char threshold3, unsigned	
char color1, unsigned char	
color2)	
void BlackWhite(char*	黑白化, input 是输入的原图像, output 是
input, char* output)	输出的黑白图像。支持 24 位 BMP 图像。
void Underexposure(char*	图像欠曝光, input 是输入的原图像,
input, char* output)	output 是输出的欠曝光图像。支持 24 位
	BMP 图像。
void Overexposure(char*	图像过曝光, input 是输入的原图像,
input, char* output)	output 是输出的过曝光图像。支持 24 位
	BMP 图像。
void Nostalgia(char*	怀旧滤镜, input 和 Mask 都是输入的文件
input, char* Mask, char*	名,Mask 是褶皱图像路径,ratio=100。
output, int ratio)	
void GammaTransform(char*	伽马变换, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void GrayScale(char*	灰度化, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void	灰度图二值化, bit 用于设定位数, 如
GrayImageBinarization(char*	bit=8 ; threshold 是阈值,如
input, char* output, int	threshold=200。支持 8 位 BMP 图像。
bit, int threshold)	
void GreyPesudoColor(char*	灰度图伪彩色化, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
void HoughTransform(char*	霍夫变换, input 是输入的 RAW 文件, output
input, char* output, unsigned	是输出的 RAS 文件,threshold=100。
char threshold)	
static void	边缘检测,参考: a=0.33, b=0.33, c=0.33。
EdgeDetectionWithoutNonmaxim	支持 24 位 BMP 图像。
um(const LPCTSTR input, const	
LPCTSTR output, double	
a, double b, double c)	
static void	参考: a=0.33, b=0.33, c=0.33。支持 24 位
NonmaximumWithoutDoubleThres	BMP 图像。
holding(const LPCTSTR input,	
const LPCTSTR output, double	

a, double b, double c)	
static void	边缘检测,参考: orank=20, oranb=80。支
CannyEdgeDetection(const	持 24 位 BMP 图像。
LPCTSTR input, const LPCTSTR	1/1 or in Dur in 18/0
<u> </u>	
output, double a, double	
b, double c, int orank, int	
oranb)	序十亦枚 分表 0 00 1 0 00 0 00 0
static void	霍夫变换,参考: a=0.33, b=0.33, c=0.33,
HoughTransform(const LPCTSTR	orank=20, oranb=80。支持 24 位 BMP 图像。
input, const LPCTSTR	
output, double a, double	
b, double c, int orank, int	
oranb)	其7世子标卷44 土井 DMO 之世
void BoxBlurBasic(string	基础方框模糊,支持 PNG 文件。
input, string output)	1
void	计算累加直方图并映射, input 是输入文件
CalculateCumulativeHistogram	名,outfile 是输出文件名。支持 24 位 BMP
Map(char* input, char*	图像。
outfile)	圆梅亚珍 • 日松)44~70. 1 79 1
void Translation(string	图像平移, input 是输入的文件, dx 和 dy
input, char* output, int	是横向及纵向的移动距离(像素),负值是
dx, int dy)	向左 / 向下移动; output 是平移操作后的
	结果文件名。支持 BMP 图像。
void Mirrored(string	镜像变换,input 是输入的文件,output 是
input, char* output, char	镜像操作后的结果文件名, axis 是镜像变
axis)	换的方向(以X或Y表示)。支持BMP图像。
void Sheared(string	错切变换,input 是输入的文件,output 是
input, char* output, char	错切操作后的结果文件名, axis 和 Coef 分
axis, double Coef)	别是错切变换的方向(以 X 或 Y 表示)和错
	切系数,负值是向左 / 向下偏移。支持 BMP
	图像。
void Scaled(string	缩放操作, input 是输入的文件, output 是
input, char* output, double	缩放操作后的结果文件名, cx 和 cy 分别是
cx, double cy)	横向及纵向的缩放系数,系数大于1表示拉
	伸,小于1表示压缩。支持BMP图像。
void Rotated1(string	图像旋转, input 是输入的文件, output 是
input, char* output, double	图像旋转后的结果文件名, angle 是旋转角
angle)	度,弧度制。支持 BMP 图像。
void SaltNoise(char*	添加椒盐噪声,a和b是噪声相关参数,如
input, char* output, int a, int	a=3, b=3; c 和 d 是颜色相关参数,如 c=0,
b, int c, int d)	d=255。支持 8 位 BMP 图像。
void CrossProcess(char*	交叉冲印滤镜,参考: ratio=100。
input, char* output, int	
ratio)	

void Conversion8(unsigned	unsigned char**转 short**, output 用于
char** input, short** output)	保存结果(与 input 大小相同)。
void Conversion8(short**	short**转 unsigned char**, output 用于
· ·	_
input, unsigned char**	保存结果(与 input 大小相同)。
output)	
void Conversion8 (unsigned	unsigned char**转 int**, output 用于保
char** input, int** output)	存结果(与 input 大小相同)。
void Conversion8(int**	int**转 unsigned char**, output 用于保
input, unsigned char**	存结果(与 input 大小相同)。
output)	
void Conversion8(unsigned	unsigned char**转 unsigned int**,
char** input, unsigned int**	output 用于保存结果(与 input 大小相同)。
output)	
void Conversion8 (unsigned	unsigned int**转 unsigned char**,
int** input, unsigned char**	output 用于保存结果(与 input 大小相同)。
output)	
void Conversion8 (unsigned	unsigned char**转 float **, output 用于
char** input, float** output)	保存结果(与 input 大小相同)。
void Conversion8(float**	float **转 unsigned char**, output 用于
input, unsigned char**	保存结果(与 input 大小相同)。
output)	
void Conversion8(unsigned	unsigned char**转double **, output用
char** input, double**	于保存结果(与 input 大小相同)。
output)	
void Conversion8(double**	double **转 unsigned char**, output用
input, unsigned char**	于保存结果(与 input 大小相同)。
output)	
void Conversion8(unsigned	unsigned char**转 char **, output 用于
char** input, char** output)	保存结果(与 input 大小相同)。
void Conversion8(char**	char **转 unsigned char**, output 用于
input, unsigned char**	保存结果(与 input 大小相同)。
output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatshort **, output 用于
input,BMPMatshort** output)	保存结果(与 input 大小相同)。
void	BMPMatshort **转 BMPMat **, output 用于
Conversion24(BMPMatshort**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatint **, output 用于保
input, BMPMatint** output)	存结果(与 input 大小相同)。
void	BMPMatint **转 BMPMat **, output 用于保
Conversion24(BMPMatint**	存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatfloat **, output 用于
input, BMPMatfloat** output)	保存结果(与 input 大小相同)。
· · · · · · · · · · · · · · · · ·	NI-14 6HVIA / A

void	BMPMatfloat **转 BMPMat **, output 用于
Conversion24(BMPMatfloat** input, BMPMat** output)	保存结果(与 input 大小相同)。
void Conversion24 (BMPMat**	BMPMat **转BMPMatdouble **, output用
input, BMPMatdouble** output)	于保存结果(与 input 大小相同)。
void	BMPMatdouble **转 BMPMat **, output 用
Conversion24(BMPMatdouble**	于保存结果(与 input 大小相同)。
input, BMPMat** output)	1 William () Imput X/1, MIII 1/2
void Conversion24(BMPMat**	BMPMat **转 BMPMatchar **, output 用于
input, BMPMatchar** output)	保存结果(与 input 大小相同)。
void	BMPMatchar **转 BMPMat **, output 用于
Conversion24(BMPMatchar**	保存结果(与 input 大小相同)。
input, BMPMat** output)	N. 1. 2 Tuber 2 (1 dul 1)
void Conversion32(BMPMat**	BMPMat **转 BMPMatshort **, output 用于
input, BMPMatshort** output)	保存结果(与 input 大小相同)。
void	BMPMatshort **转 BMPMat **, output 用于
Conversion32(BMPMatshort**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion32(BMPMat**	BMPMat **转 BMPMatint **, output 用于保
input, BMPMatint** output)	存结果(与 input 大小相同)。
void	BMPMatint **转 BMPMat **, output 用于保
Conversion32(BMPMatint**	存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion32(BMPMat**	BMPMat **转 BMPMatfloat **, output 用于
input, BMPMatfloat** output)	保存结果(与 input 大小相同)。
void	BMPMatfloat **转 BMPMat **, output 用于
Conversion32(BMPMatfloat**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion32(BMPMat**	BMPMat **转BMPMatdouble **, output用
input, BMPMatdouble** output)	于保存结果(与 input 大小相同)。
void	BMPMatdouble **转BMPMat **, output用
Conversion32(BMPMatdouble**	于保存结果(与 input 大小相同)。
input,BMPMat** output)	
void Conversion32(BMPMat**	BMPMat **转 BMPMatchar **, output 用于
input,BMPMatchar** output)	保存结果(与 input 大小相同)。
void	BMPMatchar **转 BMPMat **, output 用于
Conversion32(BMPMatchar**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void MeanFiltering(char*	均值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持8位BMP图像。
void MeanFlteringl(char*	均值滤波,input 是输入文件名,output 是
input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
void KapoorAlgorithm(char*	卡普尔算法,input 是输入文件名,output
input, char* output, int	是输出文件名。BeforeThreshold 是初始阈

D-£Tl 1 11\	店 如 D. C The 1 1 1-150 士社 0 户 DID
BeforeThreshold)	值,如 BeforeThreshold=150。支持 8 位 BMP 图像。
void OpenOperation(char*	开运算, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 4 位 BMP 图像。
void Diffusion(char*	扩散滤镜, 参考: ratio=90。
input, char* output, int	
ratio)	
void LapulasFiltering(char*	拉普拉斯滤波, readPath 是原图像,
readPath, char*	writePath 是处理后的图像文件名。支持 8
writePath, float	位 BMP 图像。
CoefArray[9], float coef)	各参数参考值:
	定义*3 的模板 (拉普拉斯):
	float
	CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0
	f, 2. 0f, 1. 0f, 2. 0f, 1. 0f};
	定义模板前乘的系数(拉普拉斯):
	float coef=(float) (1.0/16.0);
void ImageFiltering(char*	图像滤波, input 是输入文件名, output 是
input, char* output, float	输出文件名。kernel 是模糊内核。支持 24
kerne1[3][3])	位 BMP 图像。
void ComicStrip(char*	连环画滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void	亮度对比度调节,参考: brightness=-30,
BrightnessAdjustment1(char*	contrast=100°
input, char* output, int	
brightness, int contrast)	
void	亮度对比度调节,参考: brightness=-30,
BrightnessAdjustment2(char*	contrast=100.
input, char* output, int	
brightness, int contrast)	
void	零填充与对称扩展,支持 8 位和 24 位 BMP
ZeroFillingSymmetricExtensio	图像。
n(char* input, char* output)	
void PopArtStyle(char*	流行艺术风滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void LightLeakage(char*	漏光滤镜, input 和 Mask 都是输入的图像
input, char* Mask, char*	名,Mask 是漏光模板图像,ratio=90。
output, int ratio)	
void LinearFiltering(char*	线性滤波, input 是输入文件名, output 是
input, char* output, short	输出文件名。支持 8 位 BMP 图像。
average[3][3])	参考模板:
	short average[3][3] = $\{\{1, 2, 1\},$

	(0, 1, 0)
	$\{2, 4, 2\}, $ $\{1, 2, 1\}\};$
void MedianFiltering(char*	中值滤波, input 是输入文件名, output 是
_	
input, char* output, short	输出文件名。支持 8 位 BMP 图像。
average[3][3])	参考模板:
	short average[3][3] = $\{\{1, 2, 1\},$
	$\{2, 4, 2\},\$
	$\{1, 2, 1\}\};$
void	锐化滤波, input 是输入文件名, output 是
SharpeningFiltering(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output, short	参考模板:
average[3][3], short	short average[3][3] = $\{\{1, 2, 1\},$
sharpen[3][3])	$\{2, 4, 2\},$
	$\{1, 2, 1\}\};$
	short sharpen[3][3] = $\{\{-1, -1, -1\},$
	{-1, 8, -1},
	$\{-1, -1, -1\}\};$
void	梯度锐化, input 是输入文件名, output 是
GradientSharpening(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output, short	参考模板:
average[3][3], short	short average[3][3] = {{1, 2, 1},
soble1[3][3], short	$\{2, 4, 2\},$
soble2[3][3])	$\{1, 2, 1\}\};$
	short soble1[3][3] = $\{\{-1, -2, -1\},$
	{0, 0, 0},
	$\{1, 2, 1\}\};$
	short $soble2[3][3] = \{\{-1, 0, 1\},$
	$\{-2, 0, 2\},\$
	$\{-1, 0, 1\}\};$
void	算术平均滤波器, input 是输入文件名,
ArithmeticMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	output Ammay 11 a 2 N a 12 Dim Black
	日气亚均滤油器 : 日於)文件名
void	几何平均滤波器,input 是输入文件名,
GeometricMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	调和平均滤波器, input 是输入文件名,
HarmonicMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	反调和平均滤波器, input 是输入文件名,
ContraHarmonicMeanFilter(cha	output 是输出文件名。支持 8 位 BMP 图像。
r* input, char* output)	\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
void Filter (char*	滤波, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 8 位 BMP 图像。
void GreenAtmosphere(char*	绿色氛围。

• • • •	1
input, char* output)	
void GreenAtmosphere(char*	绿色氛围。支持 24 位 BMP 图像。
input, char* output, int	
width, int height, double a)	
void BMPtoJPG(char*	BMP 图像转为 JPG 图像。支持 24 位 BMP 图
input, char* output, int a)	像,尺寸必须是 8 的倍数,a 代表文件压缩
	程度,数字越大,压缩后的文件体积越小,
	如 a=100。
void Mosaic(char*	马赛克化图像, input 是输入文件名,
input, char* output, int x)	output 是输出文件名。x 是马赛克处理的块
	的大小。支持 24 位 BMP 图像。
void MosaicFilter(char*	马赛克滤镜,参考: ratio=50。
input, char* output, int	
ratio)	リロケ 左 国
void MuddyAtmosphere(char*	泥色氛围。
input, char* output)	BH/III/ I DEAN AND DE
void Expansion(char*	膨胀, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 4 位 BMP 图像。
void SmoothSharpen(char*	平滑, input 是输入文件名, output 是输出
input, char* output, int	文件名。Template 是平滑模板,均一化处
Template[3][3], int	理, coefficient1 = 9。支持24位BMP图
coefficient)	像。
void	高斯平滑, input 是输入文件名, output 是
GaussSmoothSharpen(char*	输出文件名。Template 是高斯平滑模板,
input, char* output, int	coefficient=16。支持 24 位 BMP 图像。
Template[3][3], int	
coefficient)	
void SobelSharpen(char*	
input, char* output, int	是输出文件名。Templatex 是 laplace 锐化
Templatex[3][3], int	模板,4 邻域,Templatey 是 laplace 锐化模
Templatey[3][3], int	板,8 邻域, coefficient1 = 9,
coefficient1, int	coefficient2 = 16。支持24位BMP图像。
coefficient2)	计体验证
void MidSmoothing(char*	中值滤波器,input 是输入文件名,output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void AvgSmoothing(char*	均值滤波器,input 是输入文件名,output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void Averaging (char*	图像平均化,input 是输入文件名,output
input1, char* input2, char*	是输出文件名。a 是平均化相关参数,如
input3, char* output, int a)	a=3。支持 8 位 BMP 图像。
void PlaneSlicing(char*	平面切片, input 是输入文件名, output 是 输出文件名。支持 8 位 BMD 图像
input, char* output)	输出文件名。支持 8 位 BMP 图像。 图像平移 参考, voffset==100, voffset==
void Translation(char*	图像平移,参考: xoffset=-100, yoffset=-
input, char* output, int	100。

xoffset, int yoffset)	
void	 锐化空间滤波器,input 是输入文件名,
SharpeningSpatialFiltering8(output 是输出文件名。model 是锐化模板。
char* input, char* output, int	支持8位灰度图像。
model[9])	文50世次及图像。
void PseudoGrayscale(char*	伪灰度化, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void TwoColors (char*	
·	二色化,input 是输入文件名,output 是输
input, char* output, int	出文件名。 threshold 是阈值,如 threshold=115; color1和 color2是要填
threshold, unsigned char	
color1, unsigned char color2)	充的两个颜色。支持 24 位 BMP 图像。 filename 是生成的 PNG 图像文件名; img 是
void	
PNGImageGeneration(char*	图像的像素数据,W 是图像的宽,H 是图像
filename, const unsigned char	的高,x=0 选择生成 RGB 图像,x=1 选择生
img[], unsigned W, unsigned	成 RGBA 图像。
H, int x) void MakeSphere(double	使用反射模型在正交投影下生成球体的图
V[3], double S[3], double r,	像,V是摄影机的方向, output 是输出的结
double a, double m, int ROWS,	果图像文件名,ROWS 是输出图像的行数,
int COLS, char* output)	COLS 是输出图像的列数,参考: V[3] =
	$\{0.0, 0.0, 1.0\}, S[3] = \{0.0, 0.0, 1.0\}, S[3] = \{0.0, 0.0, 0.0\}$
Mala Cabana (daubla	1.0}, r=50, a=0.5, m=1。支持 RAS 文件。
<pre>void MakeSphere(double vector_v[3], double</pre>	使用反射模型生成球体的图像,vector_v 是摄影机的方向,output 是输出的结果图
_	像文件名,ROWS 是输出图像的行数,COLS
vector_s[3], double r, double a, double m, int ROWS, int	是输出图像的列数,参考: vector_v[3] =
a, double m, int ROWS, int COLS, char* output)	{0.0, 0.0, 1.0}, vector_s[3] = {0.0,
COLS, Char* Output)	
	0.0, 1.0}, r=50, a=0.5, m=1。支持 RAS 文 件。
void	双边滤波, input 是输入文件名, output 是
BilateralFiltering(string	输出文件名。支持 24 位 BMP 图像。ssd 和
input, char* output, double	sdid 分别是空间域标准差与强度域标准
ssd, double sdid)	差。
void	五。 具有圆形结构集的双层形态侵蚀,支持 8 位
DoubleLayerErosion(char*	共有國形和构集的双层形态反照, 文持 o 位 和 24 位 BMP 图像。
input, char* output)	THE TIED IN ENDO
void	二值图像水平镜像。
BinaryImageHorizontalMirror(→四国家小「吃像。
unsigned char	
*input, unsigned char	
*output, unsigned int	
w, unsigned int h)	
void	灰阶图像水平镜像。
GrayImageHorizontalMirror(un	
orayrmagenorizonralMirror (un	

signed char *input, unsigned	
char *output, unsigned int	
w, unsigned int h)	3/4 国 // L 示 /e //
void	彩色图像水平镜像。
ColorImageHorizontalMirror(u	
nsigned char *input,unsigned	
char *output, unsigned int	
w, unsigned int h)	
void SketchFilter(char*	素描滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void Zoom(char* input, char*	缩放,参考: scaleX=5, scaleY=5,
output, float scaleX, float	interpolation=0或interpolation=1。
scaleY, int interpolation)	
int Equal(char* input1, char*	若比对图像的梯度幅相似性偏差值等于 c
input2, double 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*	若比对图像的梯度幅相似性偏差值大于 c
input1, char* input2, double	则通过。input1 和 input2 是要比对的两个
c)	图像。c 是参考的阈值。支持 24 位 BMP 图
	像。
int LessThan(char*	若比对图像的梯度幅相似性偏差值小于 c
input1, char* input2, double	则通过。input1和 input2是要比对的两个
c)	图像。c 是参考的阈值。支持 24 位 BMP 图
	像。
double GMSD(char* input1,	求两幅图像的梯度幅相似性偏差值并返回
char* input2)	结果。input1和 input2是要比对的两个图
char* input2)	结果。
void AddGaussNoise(char*	添加高斯噪声, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
void	添加椒盐噪声, input 是输入文件名,
AddSaltPepperNoise(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void ChannelSeparation(char*	通道分离,input 是输入文件名,Routput
input, char* Routput, char*	是红色通道图像,Goutput 是绿色通道图
Goutput, char* Boutput)	像,Boutput 是绿色通道图像。支持 24 位
	BMP 图像。
void PatternMethod(char*	图案法, input 是输入文件名, output 是输
input, char* output, unsigned	出文件名。Template 是模板数组。支持 8 位
char Template[8][8])	BMP 图像。
void	图层算法, input 是基底图层图像,
LayerAlgorithm(char*input,ch	inputMix 是混合图层图像,参考:
ar* inputMix, char*	alpha=50, blendModel=26.

output, int alpha, int	blendModel 的取值对应的模式如下:
blendModel)	1 典型
	2 溶解
	3 暗化
	4 多层
	5 颜色加深模式
	6 线性加深
	7 暗调
	8 亮化
	9 遮盖
	10 颜色减淡模式
	11 线性减淡
	12 浅色
	13 叠加
	14 柔光模式
	15 强光模式
	16 抱光模式
	17 线性光模式
	11
	19 强混合模式
	20 差分
	21 排除模式
	22 减运算
	23 图像分割
	24 色相模式
	25 色饱和
	26 着色
	27 亮度模式
void	图像有损压缩,input 是待压缩的 BMP 文件
BMP24LossyCompression(char*	名, output 是有损压缩后输出的文件名。支
input, char* output)	持 24 位 BMP 图像。
void	图像有损解压,input 是待解压的文件名,
BMP24LossyDecompression(char	output 是输出解压后的 BMP 文件名。支持
* input, char* output)	24 位 BMP 图像。
void	图像无损压缩, input 是待压缩的 BMP 文件
BMP24Loss1essCompression(cha	名, output 是无损压缩后输出的文件名。支
r* input, char* output)	持 24 位 BMP 图像。
void	图像无损解压,input 是待解压的文件名,
BMP24Loss1essDecompression(c	output 是输出解压后的 BMP 文件名。支持
har* input, char* output)	24 位 BMP 图像。
void	图像变色, input 是输入文件名, output 是
ImageDiscoloration(char*	输出文件名。如: a=0.2126, b=0.7152,
input, char* output, double	c=0.0722。支持 24 位 BMP 图像。
a, double b, double c)	

unsigned char** HorizontalConcavity(unsigned char** input, int RANGE, int height, int width)	图像变形之水平内凹,返回处理结果。参考: RANGE=400。
unsigned char** HorizontalConvexity(unsigned char** input, int RANGE, int height, int width)	图像变形之水平外凸,返回处理结果。参考: RANGE=400。
unsigned char** TrapezoidalDeformation(unsigned char** input,intheight,intwidth,double k)	图像变形之梯形形变,返回处理结果。参考: k=0.3。
unsigned char** TriangularDeformation(unsign ed char** input,int height,int width,double k)	图像变形之三角形形变,返回处理结果。参考: k=0.5。
unsigned char** SDeformation(unsigned char** input, int height, int width, int RANGE)	图像变形之 S 形变,返回处理结果。参考: RANGE=450。
<pre>int LsdLineDetector(unsigned char *src, int w, int h, float scaleX, float scaleY, boundingbox_t bbox, std::vector<line_float_t> &lines)</line_float_t></pre>	LSD 直线检测器。 [in] src: 图像,单通道 [in] w: 宽 [in] h: 高 [in] scaleX: X 轴上的缩小因子 [in] scaleY: Y 轴上的缩小因子 [in] bbox: 要检测的边界框 [in/out] lines: 结果 return: 0:ok; 1:error 需引入以下结构体: typedef struct { int x; int y; int width; int height; } boundingbox_t; typedef struct { float startx; float starty; float endx; float endy; } line_float_t;

```
int
                          边缘划线检测器。
                                            图像,单通道
EdgeDrawingLineDetector (unsi
                          [in]
                                   src:
gned char *src, int w, int
                                                 宽
                          [in]
                                   w:
h, float scaleX, float scaleY,
                                                 高
                          [in]
                                   h:
boundingbox t
                          [in]
                                 scaleX:
                                           X轴上的缩小因子
                    bbox,
std::vector<line float t>
                          [in]
                                 scaleY:
                                           Y轴上的缩小因子
                                            要检测的边界框
&lines)
                          [in]
                                  bbox:
                          [in/out] lines:
                                             结果
                          return:
                                            0:ok; 1:error
                          需引入以下结构体:
                          typedef struct
                             int x;
                             int y;
                             int width;
                             int height;
                          }boundingbox t;
                          typedef struct
                             float startx;
                             float starty;
                             float endx;
                             float endy;
                          }line float t;
                          传播滤波器。
int
                                               输入图像
PropagatedFilter1 (unsigned
                          [in]
                                  src:
                          [in]
                                                引导图像
char
     *src,
           unsigned
                                    guidance:
                     char
                                                输出图像
*guidance,
           unsigned
                     char
                          [in/out]]
                                    dst:
*dst, int w, int h, int c, int
                          [in]
                                                 宽
                                  W:
                          [in]
                                                 高
r, float
          sigma s,
                    float
                                  h:
                          [in]
                                      图像通道,仅 c=1 或 c=3
sigma r)
                                   c:
                          [in]
                                  r:
                                          局部窗口半径
                                    sigma s:坐标空间中的滤波器
                          [in]
                          西格玛。参数的值越大,意味着只要颜色足
                          够接近, 更远的像素就会相互影响(请参见
                          sigmaColor)。当 d>0 时,它指定邻域大小,
                          而不考虑 sigmaSpace。否则, d 与
                          sigmaSpace 成比例。
                                    sigma r:颜色空间中的滤波器
                          [in]
                          西格玛。该参数的值越大, 意味着像素邻域
                           (请参见 sigmaSpace) 内更远的颜色将混
                          合在一起,从而产生更大的半等色区域。
                                      0:ok; 1:error
                          @return:
                          传播滤波器。
int
```

PropagatedFilter2(unsigned [in] src:
] guidance: 引导图像
	/out]] dst: 输出图像
*dst, int w, int h, int c, int [in	
] w: 宽] h: 高
] c: 图像通道,仅 c=1 或 c=3
-]
	sigma_s:坐标空间中的滤波器
	好。参数的值越大,意味着只要颜色足。 第155 第156 第156 图 图 图 图 图 图 图 图 图 图 图 图 图 图 图 图 图 图 图
	接近, 更远的像素就会相互影响(请参见
	maColor)。当 d>0 时,它指定邻域大小,
	不考虑 sigmaSpace。否则,d 与
	maSpace 成比例。
] sigma_r:颜色空间中的滤波器
	3玛。该参数的值越大, 意味着像素邻域
	f参见 sigmaSpace) 内更远的颜色将混
	E一起,从而产生更大的半等色区域。
	turn: 0:ok; 1:error
_	江 滤波。
char *src, unsigned char [in	
*dst, int w, int h, int c, int [in	/out] dst: 输出图像,单通道
r)] w: 宽
[in] h: 高
[in] c: 图像通道,仅 c=1
[in] r: 局部窗口半径
ret	urn: 0:ok; 1:error
	至滤波。
BoxfilterFilter1(unsigned [in	
char *src, unsigned char [in	/out] dst: 输出图像,单通道
*dst, int w, int h, int c, int [in] w: 宽
r)] h: 高
[in] c: 图像通道,仅 c=1
[in] r: 局部窗口半径
ret	urn: 0:ok; 1:error
int 快返	导向滤波
fast_guided_filter(unsigned [in	,,,,
char *src, unsigned char [in] guidance: 引导图像,单通道
	/out] dst: 输出图像,单通道
*dst, int w, int h, int c, int [in	
r, float rp, float sr, float [in] h: 高
_scale) [in] c: 图像通道,仅 c=1
[in] r: 局部窗口半径
[in] rp: 正则化参数; eps
[in] sr: 二次采样率, sr>1: 向下

	/克士/
	缩放, 0 <sr<1: th="" 向上缩放<=""></sr<1:>
	如果正则化,_scale = 1; 如果不正则化,
	_scale = 255*255
	return: 0:ok; 1:error
	eg: $r = 4$, $(try sr = r/4 to sr=r)$, $(try $
	rp=0.1^2, 0.2^2, 0.4^2)
	try: (src, guidance, dst, w, h, 1, 4, 0.01, 4,
	255*255)
	condition: $(MIN(w, h) / sr) > 1$
	condition: (int) (r / sr + 0.5f) >= 1
int	快速导向滤波
int	
fast_guided_filter1(unsigned	
char *src, unsigned char	
*guidance, unsigned char	[in/out] dst: 输出图像,单通道
*dst, int w, int h, int c, int	[in] w: 宽
r, float rp, float sr, float	[in] h: 高
_scale)	[in] c: 图像通道,仅 c=1
	[in] r: 局部窗口半径
	[in] rp: 正则化参数: eps
	[in] sr: 二次采样率, sr>1: 向下
	缩放,0 <sr<1: th="" 向上缩放<=""></sr<1:>
	如果正则化,_scale = 1; 如果不正则化,
	scale = 255*255
	return: 0:ok; 1:error
	eg: $r = 4$, (try $sr = r/4$ to $sr=r$), (try
	rp=0.1^2, 0.2^2, 0.4^2)
	try: (src, guidance, dst, w, h, 1, 4, 0.01, 4,
	255*255)
	condition: $(MIN(w, h) / sr) > 1$
	condition: (int) $(r / sr + 0.5f) >= 1$
int	霍夫线探测器。
HoughLineDetector (unsigned	[in] src: 图像,单通道
char *src, int w, int h, float	[in] w: 宽
scaleX, float scaleY, float	[in] h: 高
CannyLowThresh, float	[in] scaleX: X 轴上的缩小因子
CannyHighThresh, float	[in] scaleY: Y轴上的缩小因子
HoughRho, float HoughTheta,	[in] CannyLowThresh: canny 算子中迟滞
float MinThetaLinelength,	过程的低阈值
float MaxThetaGap, int	[in] CannyHighThresh: canny 算子中迟滞
HoughThresh, HOUGH LINE TYPE	过程的高阈值
CODE type, boundingbox t	[in] HoughRho: 累加器的距离分辨率(以
_ ••• / • =	【III】nougliklio.素加益的距离分辨率(以 像素为单位)
bbox,	
std::vector <line_float_t></line_float_t>	[in] HoughTheta: 累加器的角度分辨率
&lines)	(弧度)

[in] MinThetaLinelength: 标准:对于 标准和多尺度 hough 变换, 检查线条的最小 角度 传播能力:最小线路长度。小于的线段被拒 绝 [in] 标准:对于标准和 MaxThetaGap: 多尺度 hough 变换,检查线条的最大角度 基于概率的:连接同一条线上的点之间允许 的最大间隙 [in] HoughThresh: 累加器阈值参数。只有 那些获得足够选票的行才会返回(>阈值) hough 线方法: [in] type: hough_line_STANDARD 或 hough_line PROBABILISTIC [in] 要检测的边界框 bbox: [in/out] lines: 结果 0:ok; 1:error return type: HOUGH LINE STANDARD: 标 准 hough 线算法 HOUGH LINE PROBABILISTIC: 概率 hough 线算法 当 HOUGH LINE STANDARD 运行时,线点可能 是图像坐标之外的位置 标准: try (src, w, h, scalex, scaley, 70, 150, 1, PI/180. 0. 100. HOUGH_LINE_STANDARD, bbox, line) 于 概 率 的 trv (src, w, h, scalex, scaley, 70, 150, 1, PI/180,30, 80, HOUGH LINE STANDARD, bbox, line). 需引入以下结构体: typedef enum HOUGH LINE TYPE CODE HOUGH LINE STANDARD = 0,//standad hough line HOUGH LINE PROBABILISTIC = 1, //probabilistic hough line } HOUGH LINE TYPE CODE;

```
typedef struct
                            int x;
                            int y;
                            int width;
                            int height;
                         }boundingbox t;
                         typedef struct
                            float startx;
                            float starty;
                            float endx;
                            float endy;
                         }line float t;
                         快速双边滤波器单通道。
void
                                      输入图像,单通道
fast bilateral filter singl
                         [in]
                                src:
                               guidance: 引导图像,单通道
                         [in]
echannel (unsigned char *src,
         char
                                       输出图像,单通道
unsigned
               *guidance,
                         [in/out]
                                 dst:
unsigned char *dst, int w,
                         [in]
                                  w:
                                        宽
int h, float sigma s, float
                         [in]
                                        高
                                  h:
sigma_r, float _scale)
                               sigma s: 坐标空间中的滤波器西
                         [in]
                         格玛。参数的值越大,意味着只要颜色足够
                         接近,更远的像素就会相互影响(请参见
                         sigmaColor)。当 d>0 时,它指定邻域大小,
                         而不考虑 sigmaSpace。否则, d 与
                         sigmaSpace 成比例。
                               sigma r: 颜色空间中的滤波器西
                         [in]
                         格玛。该参数的值越大,意味着像素邻域(请
                         参见 sigmaSpace)内更远的颜色将混合在
                         一起,从而产生更大的半等色区域。
                         如果正则化, _scale = 1; 如果不正则化,
                         scale = 255*255
                         return:
                                   0:ok; 1:error
                         快速双边滤波器单通道。
int
                                      输入图像,单通道
fast bilateral filter single
                         [in]
                                src:
                         [in]
                               guidance: 引导图像,单通道
channel (unsigned char *src,
                                       输出图像,单通道
         char
                         [in/out]
                                 dst:
unsigned
               *guidance,
                         [in]
                                        宽
unsigned char *dst, int w,
                                  w:
int h, int c, float sigma s,
                         [in]
                                  h:
                                        高
float sigma_r, float _scale)
                                       图像通道,仅 c=1
                         [in]
                                  c:
                         [in]
                               sigma s: 坐标空间中的滤波器西
                         格玛。参数的值越大, 意味着只要颜色足够
                         接近,更远的像素就会相互影响(请参见
                         sigmaColor)。当 d>0 时,它指定邻域大小,
```

而不考虑 sigmaSpace。否则, d 与 sigmaSpace 成比例。 sigma r: 颜色空间中的滤波器西 [in] 格玛。该参数的值越大,意味着像素邻域(请 参见 sigmaSpace) 内更远的颜色将混合在 一起,从而产生更大的半等色区域。 如果正则化, _scale = 1; 如果不正则化, scale = 255*2550:ok; 1:error return: 快速双边滤波器 RGB 通道。 void fast bilateral filter color [in] src: 输入图像, RGB 通道 dst: 输出图像, RGB 通道 (unsigned char *src, unsigned [in/out] char *dst, int w, [in] 宽 int h, w:高 float sigma s, float [in] h: sigma s: 坐标空间中的滤波器西 sigma_r, float _scale) [in] 格玛。参数的值越大,意味着只要颜色足够 接近, 更远的像素就会相互影响(请参见 sigmaColor)。当 d>0 时,它指定邻域大小, 而不考虑 sigmaSpace。否则, d 与 sigmaSpace 成比例。 sigma r: 颜色空间中的滤波器西 [in] 格玛。该参数的值越大,意味着像素邻域(请 参见 sigmaSpace) 内更远的颜色将混合在 一起,从而产生更大的半等色区域。 如果正则化, scale = 1; 如果不正则化, scale = 255*255return: 0:ok; 1:error int 快速双边滤波器 RGB 通道。 src: 输入图像, RGB 通道 fast bilateral filter color([in] dst: 输出图像, RGB 通道 unsigned char *src, unsigned [in/out] char *dst, int w, int h, int [in] 宽 w:float sigma s, float [in] h: 高 С, sigma_r, float _scale) [in] c:图像通道,仅 c=3 [in] sigma s: 坐标空间中的滤波器西 格玛。参数的值越大,意味着只要颜色足够 接近,更远的像素就会相互影响(请参见 sigmaColor)。当 d>0 时,它指定邻域大小, 而不考虑 sigmaSpace。否则,d 与 sigmaSpace 成比例。 [in] sigma r: 颜色空间中的滤波器西 格玛。该参数的值越大,意味着像素邻域(请 参见 sigmaSpace) 内更远的颜色将混合在 一起,从而产生更大的半等色区域。 如果正则化, _scale = 1; 如果不正则化,

	scale = 255*255
	return: 0:ok; 1:error
int	快速双边滤波器。
FastBilateralFilter(unsigned	[in] src: 輸入图像
. 9	
char *src, unsigned char	
*guidance, unsigned char	道,只有单个通道有效
*dst, int w, int h, int c,	[in/out] dst: 输出图像
float sigma_s, float	
sigma_r,float _scale)	[in] h: 高
	[in] c: 图像通道,仅 c=1 或 c=3
	[in] sigma_s: 坐标空间中的滤波
	器西格玛。参数的值越大,意味着只要颜色
	足够接近, 更远的像素就会相互影响(请参
	见 sigmaColor)。当 d>0 时,它指定邻域大
	小,而不考虑 sigmaSpace。否则,d 与
	sigmaSpace 成比例。
	[in] sigma_r: 颜色空间中的滤波器
	西格玛。该参数的值越大,意味着像素邻域
	(请参见 sigmaSpace) 内更远的颜色将混
	合在一起,从而产生更大的半等色区域。
	如果正则化,_scale = 1; 如果不正则化,
	scale = 255*255
	return: 0:ok; 1:error
	如果引导为 NULL, 仍然可以获得滤色器
int	快速双边滤波器。
permutohedral_bilateral_filt	[in] src: 输入图像
er(unsigned char *src,	[in] guidance: 引导图像
unsigned char *guidance,	[in/out] dst: 输出图像
unsigned char *dst, int w, int	[in] w: 宽
h, int c, float sigma s,	lin] h: 高
float sigma r, float scale)	c: 图像通道,仅 c=1 或
5 _ ,,	c=3
	[in] sigma s: 坐标空间中的滤波器
	西格玛。参数的值越大,意味着只要颜色足
	够接近,更远的像素就会相互影响(请参见
	sigmaColor)。当 d>0 时,它指定邻域大小,
	而不考虑 sigmaSpace。否则,d 与
	sigmaSpace 成比例。
	[in] sigma r: 颜色空间中的滤波器
	[111]
	(请参见 sigmaSpace)内更远的颜色将混
	合在一起,从而产生更大的半等色区域。
	如果正则化, _scale = 1; 如果不正则化, scale = 255*255
1	I CORIO = 755%755

	return: 0:ok; 1:error
	try: (src, guidance, dst, w, h, c, 1.6f, 0.6f
	, 255*255)
void HighPassFilter(char*	高通滤波器,参考: preserve=0。
input, char* output, int	同元····································
preserve)	
void EmbossFilter(char*	 浮雕过滤器,参考: preserve=1。
input, char* output, int	11 WILL TWO HILL S. J. PICSCIVE 10
preserve)	
void SharpenFilter(char*	锐化过滤器,参考: preserve=1。
input, char* output, int	уштах (мы на 7 у 7. ртовот то то
preserve)	
void Convolution(char*	卷积,参考: w=7, preserve=1。
input, char* output, int w, int	Given the second of the second
preserve)	
void GaussianBlur(char*	高斯模糊,参考: sigma=2, preserve=1。
input, char* output, float	1447/1041/47 2 2 1 1 2 0 mm = 1
sigma, int preserve)	
void HybridImage(char*	混合图像,参考: sigma=2, preserve=1。
input1, char* input2, char*	G V
output, float sigma, int	
preserve)	
void LowFrequencyImage(char*	低频图像,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage(char*	
input, char* output, float	
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage1(char*	
input, char* output, float	
sigma, int preserve)	
void Bilateral(char*	双边滤波,参考: sigmal=3, sigma2=0.1。
input, char* output, float	
sigmal, float sigma2)	
void SkinSmooth(char*	皮肤细滑, a 是平滑级别, b 代表是否应用
input, char* output, int a, int	皮肤过滤器, a=2, b=1。
b)	
void Resizel(char*	图像模糊,w=713,h=467。
input, char* output, int w, int	
h)	TTT //r little lite
void Resize2(char*	图像模糊。
input, char* output, int w, int	

h)	
void Shift(char* input, char*	Shift函数, ch=1, v=0.1。
output, int ch, float v)	, , , , , , , , , , , , , , , , , , ,
void RGBtoHSV(char*	RGB 转 HSV。
input, char* output)	
void HSVtoRGB(char*	HSV 转 RGB。
input, char* output)	
void RGBtoLCH(char*	RGB 转 LCH。
input, char* output)	
void LCHtoRGB(char*	LCH 转 RGB。
input, char* output)	
void ImageCutting(char*	图像裁剪, input 是输入文件名, output 是
input, char* output, int	输出文件名。leftdownx,leftdowny,
leftdownx, int leftdowny, int	rightupx, rightupy 是要裁剪的矩形区域
rightupx, int rightupy)	的左下角和右上角的坐标(连续四个整数
	值,如 50 50 300 300)。支持 24 位 BMP 图
	像。
void	图像层算法。
ImageLayerAlgorithm(char*	
input, char* output)	圆梅子 Tun 孙士成儿 · 日热)之体
void	图像无 LUT 的灰度化, input 是输入文件
RGBtoGraywithoutLUT(char*	名,output 是输出文件名。支持 24 位 BMP
input, char* output)	图像。
void RGBtoGraywithLUT(char*	图像有 LUT 的灰度化, input 是输入文件 名, output 是输出文件名。支持 24 位 BMP
input, char* output)	图像。
void	分段线性变换, input 是输入文件名,
PiecewiseLinearTransform(cha	output 是输出文件名。支持 8 位 BMP 图像。
r* input, char* output)	
void PowerConvertion(char*	功率转换, input 是输入文件名, output 是
input, char* output, double	输出文件名。如: c = 1.2, g = 0.5。支持
c, double g)	8位BMP图像。
void	拉普拉斯图像增强,input 是输入文件名,
LaplacianEnhancement(char*	output 是输出文件名。如:N=1。支持 8位
input, char* output, int N, int	BMP 图像。
LaplMask[3][3])	参考模板: int Landwark[2][2] - [
	int Lap1Mask[3][3] = {
	1, -4, 1,
	0, 1, 0
	};
void Smooth(char*	平滑, input 是输入文件名, output 是输出
input, char* output)	文件名。支持8位BMP图像。
void LaplaceSmooth(char*	拉普拉斯平滑, input 是输入文件名,
T = 11 T T T T T T T T T T T T T T T T T	

input, char* output, int N, int	output 是输出文件名。如: N=1。支持 8 位
Lap1Mask[3][3])	BMP 图像。
	参考模板:
	int LaplMask[3][3] = {
	_
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void Sobel1(char*	Sobel 算子,input 是输入文件名,output
,	
input, char* output, int N, int	是输出文件名。如: N=1。支持 8 位 BMP 图
Sb1Mask1[3][3], int	像。
Sb1Mask2[3][3])	参考模板:
	int Sb1Mask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
	, ,
	};
void SobelSmooth(char*	Sobel 平滑,input 是输入文件名,output
input, char* output, int N, int	是输出文件名。如: N=1。支持 8 位 BMP 图
Sb1Mask1[3][3], int	像。
Sb1Mask2[3][3])	参考模板:
	int Sb1Mask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	, , ,
	-1, 0, 1
	};
void Multiply(char*	图像倍增化,input 是输入文件名,output
input, char* output, int N, int	是输出文件名。如: N=1。支持 8 位 BMP 图
Sb1Mask1[3][3], int	像。
Sb1Mask2[3][3], int	
LaplMask[3][3])	int Lap1Mask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	-, -, -

```
};
                                  int Sb1Mask1[3][3] = {
                                               -1, -2, -1,
                                                0, 0, 0,
                                                1, 2, 1
                                  };
                                  int Sb1Mask2[3][3] = {
                                               -1, 0, 1,
                                               -2, 0, 2,
                                               -1, 0, 1
                                  };
                               图像添加, input 是输入文件名, output 是
     Add(char* input, char*
void
output, int
                               输出文件名。如: N=1。支持 8 位 BMP 图像。
                       N, int
Sb1Mask1[3][3], int
                               参考模板:
                               int LaplMask[3][3] = {
Sb1Mask2[3][3], int
Lap1Mask[3][3])
                                               0, 1, 0,
                                                1, -4, 1,
                                                0, 1, 0
                                  };
                                  int Sb1Mask1[3][3] = {
                                               -1, -2, -1,
                                                0, 0, 0,
                                                1, 2, 1
                                  };
                                  int Sb1Mask2[3][3] = {
                                               -1, 0, 1,
                                               -2, 0, 2,
                                               -1, 0, 1
                                  };
                              功率变换, input 是输入文件名, output 是
      PowerConvertion1(char*
void
                               输出文件名。如: c = 1.2, g = 0.5, N=1。
input, char*
               output, double
                               支持 8 位 BMP 图像。
c, double
                       N, int
             g, int
Sb1Mask1[3][3], int
                               int Lap1Mask[3][3] = {
Sb1Mask2[3][3], int
                                               0, 1, 0,
                                                1, -4, 1,
Lap1Mask[3][3])
                                                0, 1, 0
                                  };
                                  int Sb1Mask1[3][3] = {
                                               -1, -2, -1,
                                                0, 0, 0,
                                                1, 2, 1
                                  } ;
                                  int Sb1Mask2[3][3] = {
                                               -1, 0, 1,
```

	-2, 0, 2,
	-1, 0, 1
	};
void BlackWhite(char*	黑白化图像, input 是输入文件名, output
input, char* output)	是输出文件名。支持 24 位 BMP 图像。
void RandomOperation(char*	随意操作, input 是输入文件名, output 是
input, char* output, unsigned	输出文件名。支持 24 位 BMP 图像。
char treshold1, unsigned char	
treshold2, unsigned char	
treshold3, unsigned char	
treshold4, unsigned char	
treshold5, unsigned char	
treshold6, unsigned char	
red, unsigned char	
green, unsigned char blue, int	
color1, int color2, int	
color3, int color4, int	
color5, int color6, int	
color7, int color8)	
void SpecialEffects1(char*	图像特效, input 是输入文件名, output 是
input, char* output, unsigned	输出文件名。支持 24 位 BMP 图像。
char red, unsigned char	
green, unsigned char blue)	
void	怀旧滤镜,支持 24 位 BMP 图像。
NostalgicFilter(BMPMat**	
input, BMPMat** output)	
void	图像放缩,支持8位BMP图像。
SizeTransformation(short**	
input, short** output, short	
height, short width, short	
out_height, short out_width)	
void ReverseColor(short**	图像反色。
input, short** output, long	
height, long width, short	
GRAY_LEVELS)	
void Logarithm(short**	对数变换,默认 c=10。
input, short** output, long	
height, long width, short c)	
void Gamma(short**	幂律(伽马)变换,默认 c=1.2。
input, short** output, long	
height, long width, double c)	
void	直方图均衡化。
HistogramEqualization(short*	
* input, short** output, long	

height, long width, short	
GRAY_LEVELS)	77 NJ (N) (1) (2) 11
void	平滑线性滤波器。
SmoothLinearFiltering(short*	
* input, short** output, long	
height, long width, short	
average[3][3])	
<pre>void MedianFiltering(short**</pre>	中值滤波器。
input, short** output, long	
height, long width)	
void Laplace(short**	拉普拉斯算子。
input, short** output, long	
height, long width, short	
sharpen[3][3])	
void Sobel(short**	Sobel 算子。
input, short** output, long	
height, long width, short	
soble1[3][3], short	
soble2[3][3])	
void DFTRead(short** input,	二维离散傅里叶变换,实部图像。
double** output, long	
height, long width)	
void DFTImaginary(short**	二维离散傅里叶变换,虚部图像。
input, double** output, long	
height, long width)	
void FreSpectrum(short	傅里叶变换的平移。
**input, short **output,long	
height, long width)	
void IDFT(double**	二维离散傅里叶反变换。
re_array, double**	
im_array, short** output, long	
height, long width)	
void	添加高斯噪声。
AddGaussianNoise(short**	
input, short** output, long	
height, long width)	
void	添加椒盐噪声。
AddSaltPepperNoise(short**	717
input, short** output, long	
height, long width)	
void MeanFilter(short**	均值滤波器。
input, short** output, long	S PERCONAL HER
height, long width)	
void	 几何均值滤波器,默认 product=1.0。
101u	1 11111111111111111111111111111111111

C W . D:1. (1	
GeometricMeanFilter(short**	
input, short** output, long	
height, long width, double	
product)	
void	谐波均值滤波,默认 sum=0。
HarmonicMeanFiltering(short*	
* input, short** output, long	
height, long width, double	
sum)	
void	逆谐波均值滤波,Q为滤波器的阶数,Q为
InverseHarmonicMeanFiltering	正时,消除胡椒噪声,Q为负时消除盐粒噪
(short** input, short**	声, Q=0 为算术均值滤波器, Q=-1 谐波均值
output, long height, long	滤波器,默认 Q=2。
width, int Q)	
void Threshold(short**	基本全局阈值处理方法。
input, short** output, long	
height, long width, int	
delt_t, double T)	
void OTSU(short**	Otsu 方法进行最佳全局阈值处理。
input, short** output, long	
height, long width, short	
GRAY_LEVELS)	
void	基于模板矩阵的全局相加。
MatrixGlobalAddition24(BMPMa	
t** input1,BMPMat**	
input2,BMPMat** output)	
void	基于模板矩阵的全局相减。
MatrixGlobalSubtraction24(BM	
PMat** input1,BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相乘。
MatrixGlobalMultiplication24	
(BMPMat** input1,BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相除。
MatrixGlobalDivision24(BMPMa	
t** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相加。
MatrixGlobalAddition32(BMPMa	<u> </u>
t** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相减。
MatrixGlobalSubtraction32(BM	
DII	

PMat** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相乘。
MatrixGlobalMultiplication32	
(BMPMat** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相除。
MatrixGlobalDivision32(BMPMa	
t** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相加。
MatrixGlobalAddition8(unsign	
ed char** input1, unsigned	
char** input2, unsigned	
char** output)	
void	基于模板矩阵的全局相减。
MatrixGlobalSubtraction8(uns	
igned char** input1, unsigned	
char** input2, unsigned	
char** output)	
void	基于模板矩阵的全局相乘。
MatrixGlobalMultiplication8(
unsigned char**	
input1, unsigned char**	
input2, unsigned char**	
output)	
void	基于模板矩阵的全局相除。
MatrixGlobalDivision8(unsign	
ed char** input1, unsigned	
char** input2, unsigned	
char** output)	
void	彩色图以矩形方式局部截取且其它部分填
ColorRectangleLocalSegmentat	充,(x1,y1)是矩形的左上角的坐标,
ion(char* input, char*	(x2, y2) 是矩形的右下角的坐标。
output, int x1, int y1, int	函数源代码:
x2, int y2, BMPMat color)	需引入以下头文件:
	typedef struct {
	unsigned char B;
	unsigned char G;
	unsigned char R;
	unsigned char A;
	}BMPMat;
	声明:
	unsigned char** BMPRead8(char*

```
input);
                               void
                                                 GenerateImage8(char*
                               output, unsigned char** color);
                               BMPMat** BMPRead(char* input);
                               void
                                                  GenerateImage(char*
                               output, BMPMat** color, unsigned short
                               unsigned int BMPHeight(char* input);
                               unsigned int BMPWidth(char* input);
                               参考例程:
                                   BMPMat color=\{255, 255, 255\};
                                   BMPMat**
                               input=BMPRead(inputfile);
                                   BMPMat**
                               output=BMPRead(inputfile);
                                   unsigned
                                                                   int
                               height=BMPHeight(inputfile);
                                   unsigned
                                                                   int
                               width=BMPWidth(inputfile);
                                   for (unsigned
                                                     int
                                                              i
                               0; i \leq height; i++) 
                                       for (unsigned
                                                        int
                                                               j
                               0; j \le width; j++) {
                                          output[i][j]. B=color. B;
                                          output[i][j]. G=color. G;
                                          output[i][j]. R=color. R;
                                   for (unsigned
                                                     int
                                                              i
                               y1;i \le y2;i++) {
                                       for (unsigned
                                                        int
                                                               j
                               x1; j \le x2; j++) {
                               output[i][j].B=input[i][j].B;
                               output[i][j].G=input[i][j].G;
                               output[i][j]. R=input[i][j]. R;
                                   }
                               GenerateImage (outputfile, output, 24);
                               灰度图以矩形方式局部截取且其它部分填
void
                               充,(x1,y1)是矩形的左上角的坐标,
GrayRectangleLocalSegmentati
```

```
on (char*
                                (x2, y2) 是矩形的右下角的坐标。
                  input, char*
output, int
             x1, int
                       y1, int
                               函数源代码:
                                需引入以下头文件:
x2, int
          y2, unsigned
                         char
color)
                                typedef struct {
                                   unsigned char B;
                                   unsigned char G;
                                   unsigned char R;
                                   unsigned char A;
                               }BMPMat;
                               声明:
                               unsigned
                                            char**
                                                        BMPRead8(char*
                               input);
                               void
                                                 GenerateImage8(char*
                               output, unsigned char** color);
                               BMPMat** BMPRead(char* input);
                                                  GenerateImage(char*
                               void
                               output, BMPMat** color, unsigned short
                               type);
                               unsigned int BMPHeight(char* input);
                               unsigned int BMPWidth(char* input);
                                参考例程:
                                   unsigned char color=255;
                                   unsigned
                                                                char**
                                input=BMPRead8(inputfile);
                                   unsigned
                                                                char**
                               output=BMPRead8(inputfile);
                                   unsigned
                                                                   int
                               height=BMPHeight(inputfile);
                                   unsigned
                                                                   int
                               width=BMPWidth(inputfile);
                                   for (unsigned
                                                      int
                                                               i
                               0; i \leq height; i++)
                                        for (unsigned
                                                         int
                                                                j
                               0; j \leq width; j++) \{
                                          output[i][j]=color;
                                   for (unsigned
                                                      int
                                                               i
                               y1;i \le y2;i++) {
                                        for (unsigned
                                                         int
                                                                j
                               x1; j \le x2; j++) {
                                            output[i][j]=input[i][j];
```

```
GenerateImage8(outputfile, output);
                               彩色图画矩形,(x1, y1)是矩形的左上角
void
                               的坐标,(x2, y2)是矩形的右下角的坐标。
ColorDrawRectangle(char*
input, char*
                               函数源代码:
                  output, int
                               需引入以下头文件:
x1, int
           v1, int
                       x2, int
y2, BMPMat color)
                               typedef struct {
                                   unsigned char B;
                                   unsigned char G;
                                   unsigned char R;
                                   unsigned char A;
                               }BMPMat;
                               声明:
                               unsigned
                                            char**
                                                      BMPRead8(char*
                               input);
                               void
                                                GenerateImage8(char*
                               output, unsigned char** color);
                               BMPMat** BMPRead(char* input);
                               void
                                                 GenerateImage(char*
                               output, BMPMat** color, unsigned short
                               type);
                               unsigned int BMPHeight(char* input);
                               unsigned int BMPWidth(char* input);
                               参考例程:
                                   BMPMat color=\{255, 255, 255\};
                                   BMPMat**
                               input=BMPRead(inputfile);
                                   BMPMat**
                               output=BMPRead(inputfile);
                                   unsigned
                                                                  int
                               height=BMPHeight(inputfile);
                                   unsigned
                                                                  int
                               width=BMPWidth(inputfile);
                                   for (unsigned
                                                     int
                                                             i
                               0; i \leq height; i++)
                                       for (unsigned
                                                        int
                                                               j
                               0; j \le width; j++) {
                                         output[i][j]. B=color. B;
                                         output[i][j]. G=color. G;
                                         output[i][j]. R=color. R;
                                       }
                                   for (unsigned
                                                     int
                                                             i
                               0; i < height; i++) {
```

```
for (unsigned
                                                        int
                               0; j \le (idth; j++) {
                                          if(j)=x1&&j<=x2&&i==y1)
                                           output[i][j].B=color.B;
                                          output[i][j]. G=color. G;
                                          output[i][j]. R=color. R;
                                           if(j==x1\&\&i>=y1\&\&i<=y2)
                                           output[i][j]. B=color. B;
                                          output[i][j]. G=color. G;
                                          output[i][j]. R=color. R;
                                           if(j==x2\&\&i>=y1\&\&i<=y2)
                                           output[i][j]. B=color. B;
                                          output[i][j]. G=color. G;
                                          output[i][j]. R=color. R;
                                           if(j)=x1&&j<=x2&&i==y2
                                           output[i][j]. B=color. B;
                                          output[i][j]. G=color. G;
                                          output[i][j]. R=color. R;
                                       }
                                   }
                               GenerateImage (outputfile, output, 24);
void GrayDrawRectangle(char*
                               灰度图画矩形,(x1, y1)是矩形的左上角
input, char*
                   output, int
                               的坐标,(x2, y2)是矩形的右下角的坐标。
                               函数源代码:
           y1, int
                       x2, int
y2, unsigned char color)
                               需引入以下头文件:
                               typedef struct {
                                   unsigned char B;
                                   unsigned char G;
                                   unsigned char R;
                                   unsigned char A;
                               }BMPMat;
                               声明:
                                                       BMPRead8(char*
                               unsigned
                                            char**
                               input);
                               void
                                                 GenerateImage8(char*
```

```
output, unsigned char** color);
BMPMat** BMPRead(char* input);
                   GenerateImage(char*
output, BMPMat** color, unsigned short
type);
unsigned int BMPHeight(char* input);
unsigned int BMPWidth(char* input);
参考例程:
    unsigned char color=255;
    unsigned
                                  char**
input=BMPRead8(inputfile);
    unsigned
                                  char**
output=BMPRead8(inputfile);
    unsigned
                                     int
height=BMPHeight(inputfile);
    unsigned
                                     int
width=BMPWidth(inputfile);
    for (unsigned
                       int
                                i
0; i \leq height; i++) 
        for (unsigned
                          int
                                 j
0; j<width; j++) {
           output[i][j]=color;
    for (unsigned
                       int
                                i
0; i \leq height; i++) {
        for (unsigned
                          int
                                  j
0; j \le (idth; j++) 
           if(j)=x1\&\&j<=x2\&\&i==y1)
             output[i][j]=color;
             if(j==x1\&\&i>=y1\&\&i<=y2)
             output[i][j]=color;
             if(j==x2\&\&i>=y1\&\&i<=y2)
             output[i][j]=color;
             if(j)=x1\&\&j<=x2\&\&i==y2)
             output[i][j]=color;
```

	}
	}
	GenerateImage8(outputfile,output);
void Relief(BMPMat**	浮雕效果,默认 value=128。
input, BMPMat** output, int	
value)	
void Relief(unsigned char**	浮雕效果,默认 value=128。
input, unsigned char**	11 MLAX人人
1 / 9	
output, int value)	圆角岩块 雕料 10 2
void Sharpening (BMPMat**	图像锐化,默认 degree=0.3。
input, BMPMat** output, double	
degree)	
void Sharpening (unsigned	图像锐化,默认 degree=0.3。
char** input, unsigned char**	
output, double degree)	
void Soften (BMPMat**	图像柔化,默认 value=9。
input, BMPMat** output, int	
value)	
void Soften(unsigned char**	图像柔化,默认 value=9。
input, unsigned char**	
output, int value)	
void flipX(char* input, char*	X 方向翻转,支持 JPG 文件。
output)	
void flipY(char* input, char*	Y 方向翻转,支持 JPG 文件。
output)	- 74 1 4 MM 1 () 24 4 9 - 9 24 1 0
void Crop(char* input, char*	裁剪。
output, uint16_t start_x,	18421 0
uint16_t start_y, uint16_t	
new_height, uint16_t	
new_neight, uintio_t new width)	
	/定计
void Resize(char*	缩放。
input, char* output, int	
new_width, int new_height)	11. m
void Scale(char* input, char*	比例。
output, double ratio)	
void GrayscaleAvg(char*	灰度平均值。
input, char* output)	
void grayscaleLum(char*	灰度亮度。
input, char* output)	
void ColorMask(char*	彩色遮罩。
input, char* output, float	
r, float g, float b)	
void PixeLize(char*	像素化,参考: strength=2。
, , , , , , , , , , , , , , , , , , , ,	

input, char* output, int	
strength)	
void GaussianBlur(char*	高斯模糊,参考: strength=2。
input, char* output, int	同为内关的,多为:Strength Z。
strength)	
void EdgeDetection(char*	
input, char* output, double	过场和近 城中,参与:Cuttoff—115。
cutoff)	
void Sharpen(char*	锐化。
1 ,	· 坑化。
input, char* output) void GrayAVS (char*	input 是输入文件名,output 是输出文件
	名。支持8位BMP图像。
input, char* output, float	石。又行 O 位 DMF 图像。
k, float b)	古之图物练化 : 目於)文件名
Void	直方图均衡化, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图
HistogramEqualize24(char*	Output 定制出义什名。文持 24 位 BMP 图 像。
input, char* output)	版。 矩阵变换。
MatrixTransformation(char*	
input, char* output) void Binarization(char*	一店ル
	二值化。
input, char* output)	八南山萨名涌送
Void	分离出蓝色通道。
ChannelSeparation_B(char*	
input, char* output)	八南山妇女泽兴
void	分离出绿色通道。
ChannelSeparation_G(char*	
input, char* output)	八南山灰在洛泽
void	分离出红色通道。
ChannelSeparation_R(char*	
input, char* output)	5 ** *********************************
void Inverse (char*	反转。
input, char* output)	古士园构作从
void	直方图均衡化。
HistogramEqualization8(char*	
input, char* output)	江江
void Smooth (char*	平滑。
input, char* output)	Conn 質了
void CannyEdge (char*	Canny 算子。
input, char* output)	<u> </u>
void EdgeEnhance(char*	边缘增强。
input, char* output)	· 日松)之体与 · 日松山之州
void AvrFilter(char*	input 是输入文件名,output 是输出文件
input, char* output1, char*	名。如 M=21, N=1。支持 8 位 BMP 图像。
output2, int M, int N)	

void GryOppositionSSE(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持 8 位 BMP 图像。
void MedianFilter(char*	中值滤波器,input 是输入文件名,output
input, char* output, int M, int	是输出文件名。如 M=5, N=5。支持 8 位 BMP
N)	图像。
void EdgeSharpeningGry(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
void SJGryandRiceTest(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
void TextTest(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
void RedChannel(char*	生成图像的红色通道图像, input 是输入文
input, char* output)	件名,output 是输出文件名。支持 24 位 BMP
	图像。
void GreenChannel(char*	生成图像的绿色通道图像, input 是输入文
input, char* output)	件名,output 是输出文件名。支持 24 位 BMP
Imput, char. output/	图像。
void BlueChannel(char*	生成图像的蓝色通道图像, input 是输入文
input, char* output)	件名,output 是输出文件名。支持 24 位 BMP
input, char* output)	图像。
i d	1 114
void	直方图统计,input 是输入文件名,output
HistogramStatistics(char*	是输出文件名。支持 24 位 BMP 图像。
input, char* output)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization1(char*	output 是输出文件名。支持 24 位 BMP 图
input, char* output)	像。
void ReflectionRay(char*	反射线, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void MeanFiltering24(char*	均值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void MedianFiltering24(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void ZoomOutAndZoomIn(char*	缩放(双线性插值), input 是输入文件名,
input, char* output, double	output 是输出文件名。value 是放大倍数,
value)	如 value=0.5。支持 24 位 BMP 图像。
void Translation24(char*	平移, input 是输入文件名, output 是输出
input, char* output, int x, int	文件名。x 是横轴的平移量,y 是纵轴的平
y)	移量,如 x=-10, y=-30。支持 24 位 BMP 图
-	像。
void Mirror24(char*	镜像, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 24 位 BMP 图像。
void Rotate24(char*	灰转,1Nput 定输入 X 件名,Output 定输出
void Rotate24(char* input, char* output, double	旋转, input 是输入文件名, output 是输出 文件名。degree 是旋转的度数。支持 24 位
void Rotate24(char* input, char* output, double degree)	旋转, input 是制入文件名, output 是制出 文件名。degree 是旋转的度数。支持 24 位 BMP 图像。

void	给定阈值法处理图像,使图片黑白化,input
GivenThresholdMethod(char*	是输入文件名,output 是输出文件名。
input, char* output, int	threshold 是给定的阈值,如
threshold)	threshold=100。支持 24 位 BMP 图像。
void	迭代阈值法处理图像,使图片黑白化,input
IterativeThresholdMethod(cha	是输入文件名, output 是输出文件名。支持
r* input, char* output)	24 位 BMP 图像。
void	Ostu (大津法) 阈值分割, input 是输入文
OstuThresholdSegmentationMet	件名,output 是输出文件名。支持 24 位 BMP
1 1/1	图像。
hod (char* input, char* output)	
	 将伪彩图片反白,input 是输入文件名,
<u> </u>	初形图片及日,Imput 走棚八叉行名,
input, char* output)	
:1 0: 1/1	像。
void Grayl(char* input, char*	将彩色图片变成灰度图片, input 是输入文
output)	件名,output 是输出文件名。支持 24 位 BMP
:1 C W 1 1/1	图像。
void CorrectMethod(char*	正确法,input 是输入文件名,output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void	对图像分理出其中的 RGB 分量并分别保存
ChannelSeparation1(char*	为独立的图像, input 是输入文件名,
input, char* Routput, char*	Routput 是红色通道图像,Goutput 是绿色
Goutput, char* Boutput)	通道图像,Boutput 是绿色通道图像。支持
.1 D 0.1 / 1	24 位 BMP 图像。
void ReverseColor(char*	对灰度图进行反色, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
Image1* LoadImage1(char*	BMP 图像读取, input 是输入文件名。支持
input)	8位和24位BMP图像。
	返回 Image1 型数据,Image1 型数据的结构
	如下:
	typedef struct
	{
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
.1	Image1;
void SaveImage1(char*	将 Image1 型数据保存为 BMP 图像,output
output, Image1* img)	是生成的 BMP 图像文件名,img 是要保存的 图像数据,末挂 8 位和 24 位 BMD 图像
	图像数据。支持 8 位和 24 位 BMP 图像。
	Image1 型数据的结构如下:
	typedef struct
	{
	int width;

	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
	}Image1;
void	图像对比度扩展, input 是输入文件名,
ImageContrastExtension(char*	output 是输出文件名。
input, char* output, double	其中,可参考: double
m, double g1, double g2, double	m=1.5,g1=100.0,g2=200.0; m 对应斜率
a)	double a=(255.0-m*(g2-g1))/(255.0-
	(g2-g1));
	支持8位BMP图像。
void Binaryzation(char*	图像二值化, input 是输入文件名, output
input, char* output, int	是输出文件名。threshold 是将灰度值转化
threshold)	为二值的阈值,如 threshold=80。支持 24
,	位 BMP 图像。
void	全局二值化, input 是输入文件名, output
GlobalBinarization(char*	是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	自适应二值化, input 是输入文件名,
AdaptiveBinarization(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	膨胀操作, input 是输入文件名, output 是
ExpansionOperation(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	腐蚀操作, input 是输入文件名, output 是
CorrosionOperation(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void Operation1(char*	开操作, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void Closed1(char*	闭操作, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void Negativel(char*	图像反色, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void Negative(char*	图像反色, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void ImageSynthesis(char*	图像合成。
input1, char* input2, char*	
output)	
void BlackWhite(char*	黑白化,支持8位和24位BMP图像。T是
input, char* output, float	阈值,border 是边界范围,如: T=50,
T, int border)	border=0.
<pre>IMAGE Image_bmp_load(char*</pre>	加载 BMP 图片。
filename)	
void Image_bmp_save(char*	保存 BMP 图片。
<u> </u>	<u> </u>

filename, IMAGE im)	
IMAGE	缩放图片(最近邻插值法)。
TransformShapeNearest(IMAGE	加及国外(数是特面面147。
input, unsigned int newWidth,	
unsigned int newHeight)	
IMAGE	 缩放图片(双线性插值法)。
TransformShapeLinear(IMAGE	
input, unsigned int newWidth,	
unsigned int newHeight)	
IMAGE	图像的任意角度的旋转。
TransformShapeWhirl(IMAGE	
input, float angle)	
IMAGE	图像的镜像翻转。
TransformShapeUpturn(IMAGE	
input, int a)	
void	彩色图转灰度图,对于 GrayscaleMode 的
TransformColorGrayscale(IMAG	值: 1表示加权法,2表示最值法,3表示
E im, int GrayscaleMode)	均值法,4表示红色分量法,5表示绿色分
	量法,6表示蓝色分量法。
void	二值图(自定义阈值法)。
TransformColorBWDIY(IMAGE	
input, unsigned char	
Threshold)	
void	二值图(大津法 OSTU, 适用双峰直方图。)
TransformColorBWOSTU(IMAGE	
input)	
void	二值图(三角法 TRIANGLE, 适用单峰直方
TransformColorBWTRIANGLE(IMA	图。)
GE input)	
IMAGE	二值图(自适应阈值法,areaSize=25 较合
TransformColorBWAdaptive(IMA	适)
GE input, int areaSize)	
IMAGE	二值图(用二值图表示灰度变
TransformColorBWGrayscale(IM	化, areaSize=25 较合适)
AGE input, int areaSize)	
void	反色。
TransformColorOpposite(IMAGE	
input)	
IMAGE	直方图均衡化(分步计算,效果更加柔和)。
TransformColorHistogramPart(
IMAGE input)	
IMAGE	直方图均衡化(整体计算,效果更加尖锐)。
TransformColorHistogramAll(I	
MAGE input)	

IMAGE KernelsUseDIY(IMAGE	卷积操作(自定义)。
input, double* kernels, int	
areaSize, double modulus)	
IMAGE	中值滤波。
WavefilteringMedian(IMAGE	
input)	
IMAGE	高斯滤波。
WavefilteringGauss(IMAGE	高斯滤波卷积核:
input, double	double KERNELS Wave Gauss[9] =
KERNELS_Wave_Gauss[9], int	{
a, double b)	1, 2, 1,
a, double by	2, 4, 2,
	1, 2, 1
	};
IMAGE	低通滤波。
Wavefiltering_LowPass(IMAGE	
input, double* kernels)	double KERNELS Wave LowPass LP1[9] =
input, double* kerners)	
	1 / 9.0, 1 / 9.0, 1 / 9.0,
	1 / 9.0, 1 / 9.0, 1 / 9.0, 1 / 9.0,
	1 / 9.0, 1 / 9.0, 1 / 9.0
	};
	 // 低通滤波卷积核 LP2
	double KERNELS Wave LowPass LP2[9] =
	double KEKNELS_wave_Lowrass_Lf2[9] =
	1 / 10.0, 1 / 10.0, 1 / 10.0,
	1 / 10.0, 1 / 10.0, 1 / 10.0,
	1 / 10.0, 1 / 3.0, 1 / 10.0, 1 / 10.0
	};
	,
	 // 低通滤波卷积核 LP3
	double KERNELS Wave LowPass LP3[9] =
	{
	1 / 16.0, 1 / 8.0, 1 / 16.0,
	1 / 8.0, 1 / 4.0, 1 / 8.0,
	1 / 16.0, 1 / 8.0, 1 / 16.0
	};
IMAGE	高通滤波。
WavefilteringHighPass(IMAGE	
input, double* kernels)	double KERNELS Wave HighPass HP1[9] =
input, doubte. Reflicts/	{
	-1, -1, -1,
	-1, 9, -1,
	1, 0, 1,

```
-1, -1, -1
                            };
                            // 高通滤波卷积核 HP2
                            double KERNELS_Wave_HighPass_HP2[9] =
                                0, -1, 0,
                               -1, 5, -1,
                                0, -1, 0
                            };
                            // 高通滤波卷积核 HP3
                            double KERNELS Wave HighPass HP3[9] =
                                1, -2, 1,
                               -2, 5, -2,
                                1, -2,1
                            均值滤波。
IMAGE
                            // 均值滤波卷积核
Wavefiltering Average (IMAGE
                            double KERNELS_Wave_Average[25] =
input, double*
KERNELS_Wave_Average)
                              1, 1, 1, 1, 1,
                              1, 1, 1, 1, 1,
                              1, 1, 1, 1, 1,
                              1, 1, 1, 1, 1,
                              1, 1, 1, 1
IMAGE
                            差分边缘检测。
                            // 差分垂直边缘检测卷积核
EdgeDetectionDifference(IMAG
E input, double* kernels)
                            double
                            KERNELS Edge difference vertical[9] =
                                0, 0, 0,
                               -1, 1, 0,
                                0, 0, 0
                            };
                            // 差分水平边缘检测卷积核
                            doub1e
                            KERNELS Edge difference horizontal[9]
                                0, -1, 0,
```

```
0, 1, 0,
                                0, 0, 0
                             };
                             // 差分垂直和水平边缘检测卷积核
                             double KERNELS_Edge_difference_VH[9]
                                -1, 0, 0,
                                0, 1, 0,
                                0, 0, 0
IMAGE
                             Sobel 边缘检测。
                             // Sobel X 边缘检测卷积核
KernelsUseEdgeSobel(IMAGE
                             double KERNELS_Edge_Sobel_X[9] =
        double*
                  kernels1,
double* kernels2)
                                -1, 0, 1,
                               - 2, 0, 2,
                                -1, 0, 1
                             };
                             // Sobel Y边缘检测卷积核
                             double KERNELS_Edge_Sobel_Y[9] =
                                -1, -2, -1,
                                0, 0, 0,
                                1, 2, 1
IMAGE
                             Laplace 边缘检测。
                             // Laplace 边缘检测卷积核 LAP1
EdgeDetectionLaplace(IMAGE
input, double* kernels)
                             double KERNELS_Edge_Laplace_LAP1[9] =
                                0, 1, 0,
                                1, -4, 1,
                                0, 1, 0
                             };
                             // Laplace 边缘检测卷积核 LAP2
                             double KERNELS_Edge_Laplace_LAP2[9] =
                                -1, -1, -1,
                               -1, 8, -1,
                               -1, -1, -1
```

```
// Laplace 边缘检测卷积核 LAP3
                             double KERNELS Edge Laplace LAP3[9] =
                               -1, -1, -1,
                               -1, 9, -1,
                               -1, -1, -1
                             };
                             // Laplace 边缘检测卷积核 LAP4
                             double KERNELS Edge Laplace LAP4[9] =
                                1, -2, 1,
                               -2, 8, -2,
                                1, -2, 1
IMAGE
                             腐蚀。
MorphologyErosion (IMAGE
                             // 腐蚀卷积核
input, double* kernels)
                             doub1e
                             KERNELS Morphology Erosion cross[9] =
                                0, 1, 0,
                                1, 1, 1,
                                0, 1, 0
                             };
                             膨胀。
IMAGE
                             // 膨胀卷积核
MorphologyDilation(IMAGE
input, double* kernels)
                             double
                             KERNELS_Morphology_Dilation_cross[9]
                                0, 1, 0,
                                1, 1, 1,
                                0, 1, 0
     Pooling(IMAGE
IMAGE
                    input,
                             池化。
int lenght)
                             获得积分图(在此之前要保证图片是"白底
IGIMAGE
        IntegralImage(IMAGE
                             黑字")。
input)
void
        FaceDetection(char*
                             人脸检测。
input, char*
             output, double*
KERNELS Wave Average)
                             人脸检测。
IMAGE
        FaceDetection (IMAGE
input1, IMAGE
             input2, double*
                             需引入以下结构体:
```

KEDNELC W. A.	1.0
KERNELS_Wave_Average)	typedef struct tagBGRA
	unsigned char blue;
	unsigned char green;
	unsigned char red;
	unsigned char transparency;
	}BGRA, *PBGRA;
	typedef struct tagIMAGE
	{
	unsigned int w;
	unsigned int h;
	BGRA* color;
	}IMAGE, *PIMAGE;
	声明:
	<pre>IMAGE Image_bmp_load(char* filename);</pre>
	void Image bmp save(char*
	filename, IMAGE im);
	参考:
	// 用于处理
	IMAGE input2 =
	<pre>Image_bmp_load(inputfile);</pre>
	// 用于保存
	IMAGE input2=
	Image_bmp_load(inputfile);
	input2=FaceDetection(input1, input2, KE
	RNELS Wave Average);
	// 保存图片
	Image_bmp_save(outputfile, input2);
void	图像积分图。
IntegralDiagram(unsigned int	
*input, unsigned int *output,	
int width, int height)	
,	图像加密,支持 8 位、24 位和 32 位 BMP 图
0 11 '	像。inFileName 是原图图像文件名,
inFileName, char*	
outFileName, char key)	outFileName 是解密图像文件名,key 是密
	钥,如 key=255。 图像阅读 :: P:1-N 目 地 家 图 像 文 供 名
void ImageDecryption(char*	图像解密, inFileName 是加密图像文件名,
inFileName, char*	outFileName 是解密图像文件名,key 是密
outFileName, char key)	钥,如 key=255。支持 8 位、24 位和 32 位
. 1	
void	图像加解密,Key 是密钥,a=1 时执行加密,
EncryptionDecryption(char*	a=0 时执行解密。支持 24 位 BMP 图像。
input, char* output, int	
Key, int a)	

	国体指南
void Encryption(char*	图像加密, input 是输入文件名, output 是
input, char* output, int Key)	输出文件名。Key 是密钥。支持 24 位 BMP 图
	像。
void Decryption(char*	图像解密,input 是输入文件名,output 是
input, char* output, int Key)	输出文件名。Key 是密钥。支持 24 位 BMP 图
	像。
void Compress8(string	图像压缩, input 是输入文件名, output 是
input, string output)	输出文件名。支持8位BMP图像。
void Decompression(string	图像解压, input 是输入文件名, output 是
	输出文件名。支持8位BMP图像压缩后的结
input, string output)	
	果文件。
void HorizontalMirror(char*	水平镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void MirrorVertically(char*	垂直镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void XMirroring(char*	X 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void YMirroring(char*	Y 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持8位BMP图像。
void ImageConvolution(char*	图像卷积, input 是输入文件名, output 是
input, char* output, double**	输出文件名。Kernel 是卷积核,如 double
Kernel, int n, int m)	Kernel[3][3] = {{-0.225, -0.225-
Reiner, int II, int III/	·
	0. 225}, {-0. 225, 1, -0. 225}, {-0. 225, -
	0.225, -0.225}}; n 是 Kernel 的第一维的
	大小,m 是 Kernel 的第二维的大小,形如
,	Kernel[n][m]。支持 24 位 BMP 图像。
void SpatialMeanFiter(char*	空间均值过滤器,参考: radius=3。
input, char* output, int	
radius)	
void	空间中值过滤器,参考: radius=3。
SpatialMedianFiter(char*	
input, char* output, int	
radius)	
void SpatialMaxFiter(char*	空间最大过滤器,参考: radius=3。
input, char* output, int	ユニドラ4Xノベスシルの HH 1 シーフ・ I duItus 00
radius)	
void SpatialMinFiter(char*	空间最小过滤器,参考: radius=3。
input, char* output, int	
radius)	
void SpatialGaussFiter(char*	空间高斯过滤器,参考: radius=3。
input, char* output, int	
radius)	
void	空间统计滤波器,参考: radius=3, T=0.2。
SpatialStatisticalFiter(char	
<u> </u>	

* input, char* output, int	
radius, float T)	
void Mosaic (char*	刀塞古滤焙 … 和 1. 目於山園梅的密和言
	马赛克滤镜,w和h是输出图像的宽和高。
input, char* output, int w, int	支持 PNG 图像。
h)	
void FFTAmp(char*	FFT 放大器,参考: inv=false。
input, char* output, bool inv)	
void FFTPhase(char*	FFT 相位,参考: inv=false。
input, char* output, bool inv)	
void STDFT1(char*	参考: inv=false。
input, char* output, bool inv)	
void STDFT2(char*	参考: inv=false。
input, char* output, bool inv)	
void SpectrumShaping(char*	图像频域滤波, FFT 变换一相位谱,
input, char* inputMsk, char*	inputMsk 是输入的掩膜图像名。
output)	-
void Translation(char*	图像平移, input 是输入文件名, output 是
input, char* output, int x, int	输出文件名。x和y是在X轴和Y轴平移的
y, unsigned char color)	量,以右为正向,color 是平移后非原图区
j, and igned and actor,	域填充的颜色,如 color=100。支持 8 位 BMP
	图像。
void Nesting(char*	图像嵌套, Biginput 是输入的大图,
Biginput, char*	Smallinput 是输入的小图。支持 24 位 BMP
Smallinput, char* output)	图像。
void	图像去除某种像素, output 用于保存结果
CrossDenoising24(BMPMat**	(与 input 大小相同)。
input, BMPMat** output, BMPMat	(—) Imput /(/)//Impi//
threshold, BMPMat target)	
·	图像土险甘种 <u></u>
void	图像去除某种像素, output 用于保存结果
CrossDenoising8(unsigned	(与 input 大小相同)。
char** input, unsigned char**	
output, unsigned char	
threshold, unsigned char	
target)	
void	图像去污。(x1, y1)是矩形污渍区的左上角
ImageDecontamination(BMPMat*	坐标,(x2, y2)是矩形污渍区的右下角坐标。
* input, BMPMat** output, int	
x1, int y1, int x2, int y2)	
void	图像去污。(x1, y1)是矩形污渍区的左上角
ImageDecontamination(unsigne	坐标,(x2, y2)是矩形污渍区的右下角坐标。
d char** input, unsigned	
char** output, int x1, int	
y1, int x2, int y2)	
void Blend(char*	图像融合之混合化, input1 和 input2 是输

input1, char* input2, char*	入的两个要融合的图像,output 是输出文
output)	件名。支持 24 位 BMP 图像。
void Checker (char*	图像融合之棋盘化, input1 和 input2 是输
input1, char* input2, char*	入的两个要融合的图像, output 是输出文
output)	件名。支持 24 位 BMP 图像。
void Blend1(char*	图像融合之混合化, input1 和 input2 是输
input1, char* input2, char*	入的两个要融合的图像, output 是输出文
output)	件名。支持 24 位 BMP 图像。
void Checker1(char*	图像融合之棋盘化, input1 和 input2 是输
input1, char* input2, char*	入的两个要融合的图像, output 是输出文
output)	件名。支持 24 位 BMP 图像。
void ImageSharpening(char*	图像锐化, input 是输入文件名, output 是
input, char* output)	输出文件名。支持8位BMP图像。
void SharpenLaplace(char*	拉普拉斯锐化,参考: ratio=100。
input, char* output, int	
ratio)	IICM 份化
void SharpenUSM(char*	USM 锐化,参考: radius=5, amount=400,
input, char* output, int	threshold=50.
radius, int amount, int	
threshold)	4 0 1 D 2 E B L 1 2 1 1 2 1 1 2 1 2 1 2 1 2 1 2 1 2 1
void DrawRectangle(char*	在 24 位 BMP 图像上通过传入的参数画一个
input, char* output, int	矩形。input 是输入文件名,output 是输出
x1, int y1, int x2, int	文件名。(x1, y1)是矩形坐上顶点的坐标,
y2, unsigned char	(x2, y2)是矩形右下顶点的坐标; red 是矩
red, unsigned char	形线框的红色分量, green 是矩形线框的绿
green, unsigned char blue)	色分量,blue 是矩形的蓝色分量。
void GenerateBmp(unsigned	生成 BMP 图像,pData 是图像的像素数据,
char* pData, int width, int	width和height是图像的宽和高,filename
height, char* filename)	是生成的图像的文件名。
void	JPG 图像生成, filename 是生成的 JPG 图像
Jpg24ImageGeneration(char*	文件名, width 是图像的宽, height 是图像
filename, unsigned int width,	的高, img 是图像的像素数据。
unsigned int height, unsigned	
char* img)	
void	最近邻插值法去栅格, input 是输入文件
ImageScalingNearestNeighborI	名, output 是输出文件名。1x 和 1y 是长和
nterpolation(char*	宽需要缩放的倍数。支持 8 位 BMP 图像。
input, char* output, float	
1x, float 1y)	
void	双线性插值法去栅格, input 是输入文件
ImageScalingBilinearInterpol	名, output 是输出文件名。1x 和 1y 是长和
ation(char* input, char*	宽需要缩放的倍数。支持8位BMP图像。
output, float 1x, float 1y)	
void	双线性插值,input 是输入文件名,output

BilinearInterpolationScaling	是输出文件名。ExpScalValue 是期望的缩
(char* input, char*	放倍数 (允许小数)。支持 BMP 图像。
output, float ExpScalValue)	
void	最近邻插值,input 是输入文件名,output
NearestNeighborInterpolation	是输出文件名。ExpScalValue 是期望的缩
Scaling(char* input, char*	放倍数(允许小数)。支持 BMP 图像。
output, float ExpScalValue)	
void ZoomImg(unsigned char	二次线性插值图像缩放。
*input, unsigned char	
*output, int sw, int sh, int	
channels, int dw, int dh)	
void	 图像修复, output 用于保存结果 (与 input
CrossDenoising24(BMPMat**	大小相同),target 是污点像素,weight 是
	人们相同方,talget 足行点像系,weight 足 修复权重系数。
input, BMPMat** output, BMPMat	
target, BMPMatdouble weight)	圆梅极有
void	图像修复, output 用于保存结果(与 input
CrossDenoising8 (unsigned	大小相同),target 是污点像素,weight 是
char** input, unsigned char**	修复权重系数。
output, unsigned char	
target, double weight)	
void	input 是输入文件名,output 是输出文件
RotateRight90Degrees(char*	名。支持 8 位 BMP 图像,向右旋转 90 度。
input, char* output)	
void	input 是输入文件名,output 是输出文件
RotateLeft90Degrees(char*	名。支持 8 位 BMP 图像,向左旋转 90 度。
input, char* output)	
void ImageRotation(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 8 位 BMP 图像。angle 是
angle)	要旋转的角度。
void Rotation8(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 8 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标, color 是旋转后非原
color)	图区域的填充颜色。
void Rotation24(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 24 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标; red、green、blue 分
red, unsigned char	别是旋转后非原图区域要填充的颜色的红
green, unsigned char blue)	绿蓝分量。
void Rotation(char*	图像旋转, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。angle 是
angle, unsigned char color)	旋转的角度, color 是旋转后非原图区域填
•	充的颜色,如 color=100。

void Rotate(char*	图像旋转, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 BMP 图像。angle 是旋转
angle)	柳山文件石。文符 DMF 图像。dligte 定旋符 的角度。
void	灰度图像旋转 90。
imgRotate90Gray(unsigned	次反图像处存 50。
char *input, unsigned char	
*output, int sw, int sh, int *dw, int *dh)	
void	彩色图像旋转 90。
	杉巴图像灰妆 90。
imgRotate90Color(unsigned char *input, unsigned char	
1 , 0	
*output, int sw, int sh, int	
*dw, int *dh)	大队网络铁柱 970
void	灰阶图像旋转 270。
imgRotate270Gray(unsigned char *input, unsigned char	
• ,	
*output, int sw, int sh, int	
*dw, int *dh) void	
	杉巴图像灰妆 270。
imgRotate270Color (unsigned	
char *input, unsigned char	
*output, int sw, int sh, int	
*dw, int *dh)	大阪网络连柱 100 社用但左左百 <u>龄</u>)粉组
void	灰阶图像旋转 180, 结果保存在原输入数组
imgRotate180Gray(unsigned	中。
char *Img, int w, int h)	彩色图像旋转 180, 结果保存在原输入数组
void	
imgRotate180Color(unsigned	中。
char *Img, int w, int h)	彩
void imgRBExchange (unsigned	彩色图像 R、B 互换, 结果保存在原输入数 组中。
char *Img, int w, int h)	
void NoiseUniform(char*	均匀分布噪声,参考: a=0, b=0.2。
input, char* output, double	
a, double b)	
void NoiseGauss (char*	高斯噪声,参考: mean=0, delta=31。
input, char* output, float	
mean, float delta)	
void NoiseRayleigh(char*	瑞利噪声,参考: a=0, b=200。
input, char* output, float	
a, float b) void NoiseExp(char*	 指数噪声,参考: a=0.1。
1 '	1日奴 喉 戸,
input, char* output, float a)	
void NoiseImpulse(char*	椒盐噪声,参考: a=0.2, b=0.2。
input, char* output, float	

21 1)	
a, float b)	
void grayToColor(FILE*	灰色转伪彩色,input 是输入文件,output
input, FILE* output)	是输出文件。支持 8 位和 24 位 BMP 图像。
void ImageThinning(char*	图像细化, input 是输入文件名, output 是
input, char* output, char**	输出文件名。支持 4 位 BMP 图像。n 是 str
str, int n, int ml, int a, int b)	第一维的大小, m1 是第二维的大小, 形如
	str[n][m1]; a 和 b 是相关的调节参数, 可
	以为 a=3, b=5。
	参考模板:
	$[char str[6][8] = \{ \{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, $
	0, 0, }, { 255, 0, 255, 0, 0, 255, 0,
	0 },
	{ 255, 0, 255, 255, 0, 255, 0,
	255 }, { 255, 255, 255, 0, 0, 255,
	255, 255 },
	{ 255, 0, 255, 255, 0, 255, 255,
	255 }, { 0, 255, 255, 255, 255, 255,
	255, 255 } };
int	返回图像像素的最小值, filename 是输入
MinimumValueOfImagePixels(ch	的图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	
int	返回图像像素的最大值, filename 是输入
MaximumValueOfImagePixels(ch	的图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	
float	返回图像像素的均值, filename 是输入的
AverageValueOfImagePixels(ch	图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	
double	返回图像像素的标准差, filename 是输入
StandardDeviationOfImagePixe	的图像文件名。支持 8 位和 24 位 BMP 图像。
ls(char* filename)	
double EntropyOfImage(char*	返回图像的熵,支持8位和24位BMP图像。
filename)	
float*	filename 是输入的图像文件名。存储每个
CountTheFrequencyOfPixels(ch	像素的频率,像素值为 0 [~] 255, 返回值数组
ar* filename)	中的元素序号即为像素值,该序号在数组下
	的值即为这个像素的频率。支持8位和24
	位 BMP 图像。
void Rotate(char*	图 像 旋 转 。 参 考 : angle=80 ,
input, char* output, int	interpolation=0或interpolation=1。
angle, int interpolation)	
void HSV(char* input,char*	图像色调饱和度明度调节,参考: h=120,
output, int h, int s, int v)	s=60, v=20。
void OilpaintFilter(char*	油画滤镜,参考: radius=10, smooth=100。
input, char* output, int	
•	•

radius, int smooth)	
unsigned int*	 圆检测,返回圆心的坐标和半径,返回值数
CircleDetection(char* input)	组中第一个元素是圆心 X 坐标,第二个元素
circlede tection (char input)	是圆心Y坐标,第三个元素是圆半径。
void HaloFilter(char*	是例记了至你,另二十九款是图中任。 晕角滤镜,参考: ratio=100。
·	年用施锐,参 气: Tat10-100。
input, char* output, int	
ratio)	大庆丰子园 分世 1W:1/1 050
void GrayHistogram(char*	灰度直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void RedHistogram(char*	红色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void GreenHistogram(char*	绿色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void BlueHistogram(char*	蓝色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization2(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output, int	BMP 图像。imgBit 是输入图像的位数。
imgBit)	Import / Cim/ Cim/ Cim/
void	直方图均衡化, input 是输入文件名,
HistogramEqualization3(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output)	BMP 图像。
void	直方图均衡化, input 是输入文件名,
HistogramEqualization4(char*	output 是输出文件名。支持 8 位和 24 位 DMD 图像 input 目於) 文件名称 out 目
input, char* output)	BMP 图像。input 是输入文件名称,out 是
• 1	输出文件名称。
void	直方图均衡化,参考: hWidth=256,
HistogramEqualization(char*	hHeight=100。
input, char* output, int	
hWidth, int hHeight)	
void	灰度直方图, 参考: hWidth=256,
GrayHistogramEqualization(ch	hHeight=100。
ar* input, char* output, int	
hWidth, int hHeight)	
void	红色通道直方图,参考: hWidth=256,
RedHistogramEqualization(cha	hHeight=100。
r* input, char* output, int	
hWidth, int hHeight)	
void	绿色通道直方图,参考: hWidth=256,
GreenHistogramEqualization(c	hHeight=100。
J 1(*	ı <u> </u>

har* input, char* output, int	
hWidth, int hHeight)	
void	蓝色通道直方图,参考: hWidth=256,
BlueHistogramEqualization(ch	hHeight=100。
ar* input, char* output, int	
hWidth, int hHeight)	
void GrayScaleStretch(char*	灰度级拉伸,参考: hWidth=256,
input, char* output, int	hHeight=100.
hWidth, int hHeight)	
void	灰度直方图拉伸,参考: hWidth=256,
GrayHistagramStretch(char*	hHeight=100.
input, char* output, int	intergrit 100°
hWidth, int hHeight)	次
void	红色通道直方图,参考: hWidth=256,
RedHistagramStretch(char*	hHeight=100。
input, char* output, int	
hWidth, int hHeight)	
void	绿色通道直方图,参考: hWidth=256,
GreenHistagramStretch(char*	hHeight=100。
input, char* output, int	
hWidth, int hHeight)	
void	蓝色通道直方图,参考: hWidth=256,
BlueHistagramStretch(char*	hHeight=100。
input, char* output, int	
hWidth, int hHeight)	
void MedianFilteringl(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void MedianFiltering2(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
void	阈值处理, input 是输入文件名, output 是
ThresholdProcessing(char*	输出文件名。支持 8 位 BMP 图像。Threshold
input, char* output, int	- 棚山文行名。文持 6 位 bmr 図像。Infestioru - 是阈值相关参数,如 Threshold=0.001。
Threshold)	定网但4日八岁双,
· · · · · · · · · · · · · · · · · · ·	十海注加珊 in 見絵》文件名
void OTSUProcessing(char*	大津法处理, input 是输入文件名, output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void OBJtoTGA(char*	OBJ 转 TGA。
input, char* output, int	
width, int height)	如原体状刻取以原体。土土为20
void ToRIM(char* input, char*	一般图像转到 RIM 图像,支持 PNG、JPG 和
output)	TGA 图像。
void ToImage(char*	RIM 图像转到一般图像,支持 PNG、JPG 和
input, char* output, int	TGA 图像。 jpg_quality=25。
jpg_quality)	
void	将 1 位深度的单色 BMP 图片转成热敏打印

ImprimanteThermique(char* input, char* output, ARRAY3 skip_cmd, unsigned short PRINTER_TYPE_BMP, unsigned char mode, unsigned int FILE_TYPE_AD, unsigned char a, unsigned char b) void WhiteBalance(const char* input, const char* output)	机的位图打印输出,支持的热敏打印机的位图打印指令为〈strong〉ESC *〈/strong〉指令。 typedef unsigned char ARRAY3[3];参考:output="output.pbin", skip_cmd = {0x1B, 0x4A, 0x00}, PRINTER_TYPE_BMP是打印机位图打印模式, printer_Type_BMP=(0x2A1B), mode是打印机位图打印模式, mode=33, FILE_TYPE_AD是图片类型, "AD"表示广告图片,FILE_TYPE_AD=(0x4441), a=0x80, b=1。白平衡。
void Sobel(char* input, char* output, double magnScale, double threshold)	Sobel 算子, magnScale=0.35, threshold=130。支持PGM和PBM图像。
void Canny(char* input, char* output, double magnScale, double lowThreshold, double highThreshold)	Canny 算子, magnScale=0.35, lowThreshold=55, highThreshold=120。支持PGM和PBM图像。
void BlackWhite(char* input, char* output, int threshold, int background)	黑白化,threshold=100, background=0。 支持 PGM 和 PBM 图像。
void ConnectedComponents(char* input, char* output, int threshold, int background, int threshold1)	区域连通, threshold=100, background=0, threshold1=100。支持 PGM 和 PBM 图像。
void CleanImage(char* input, char* output)	清洁图像。支持 PGM 和 PBM 图像。
<pre>void NoiseImage(char* input, char* output, float probability)</pre>	噪声化图像,probability=0.1。支持 PGM 和 PBM 图像。
<pre>void HoughTransformCircle1(char* input, char* output, double sigma, int kernelSize, int scale, double gamma, double magnScale, double lowThreshold, double highThreshold, int scale1, double gamma1)</pre>	圆 检 测 。 scale1=1 , gamma1=1.0 , magnScale=0.5 , lowThreshold=85 , highThreshold=150, scale=0, gamma=1.0, sigma 和 kernelSize 用于平滑的高斯 5x5 内核, sigma=1.0, kernelSize=5。 如果 scale==0,则值保持不变,但如果它们低于 0 或高于 255,则它们分别设置为 0 或 255。 如果 scale!=0,则对值进行缩放,使得最

小值为零,最大值为255。 设置 gamma 值允许指数缩放,启用 gamma==1.0. 支持 PGM 和 PBM 图像。 void 圆检测。scale1=1, gamma1=1.0, HoughTransformCircle2(char* magnScale=0.5, lowThreshold=85 highThreshold=150, scale=0, gamma=1.0, input, char* output, int number, int minDist, double sigma 和 kernelSize 用于平滑的高斯 5x5 内核,sigma=1.0,kernelSize=5,number=10 sigma, int kernelSize, int 表示图像的目视检查有 10 个圆圈, scale, double gamma, double magnScale, double minDist=35. lowThreshold, double 如果 scale==0,则值保持不变,但如果它们 highThreshold, int 低于0或高于255,则它们分别设置为0或 scale1, double gamma1) 255。 如果 scale! =0,则对值进行缩放,使得最 小值为零,最大值为255。 设置 gamma 值允许指数缩放,启用 gamma==1.0. 支持 PGM 和 PBM 图像。 double** 圆检测。scale1=1, gamma1=1.0, HoughTransformCircle3(char* magnScale=0.5 , lowThreshold=85 input, char* output, int highThreshold=150, scale=0, gamma=1.0, sigma 和 kernelSize 用于平滑的高斯 5x5 number, int minDist, double sigma, int kernelSize, int 内核,sigma=1.0,kernelSize=5,number=10 gamma, double 表示图像的目视检查有 10 个圆圈, scale, double magnScale, double minDist=35. lowThreshold, double 如果 scale==0,则值保持不变,但如果它们 highThreshold, int 低于0或高于255,则它们分别设置为0或 scale1, double gamma1) 255。 如果 scale! =0,则对值进行缩放,使得最 小值为零,最大值为255。 设置 gamma 值允许指数缩放,启用 gamma==1.0. 返回以(vCenter, hCenter)和半径 (vRadius, hRadius) 为中心的椭圆数据, 共有 number 组数据,每组包含一个椭圆数 据,第一个元素是 vCenter,第二个元素是 hCenter,第三个元素是 vRadius,第四个元 素是 hRadius。 支持 PGM 和 PBM 图像。 形状边缘检测, CANNY THRESH4=35, void CANNY_blur4=7。支持 PNG 图像。 ShapeEdgeDetection1(char*

input, char*

char

output, unsigned

CANNY THRESH4, int

CANNY_blur4)	
void	形状边缘检测,CANNY_THRESH4=35,
ShapeEdgeDetection2(char*	CANNY_blur4=7。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH4, int	
CANNY_blur4)	
void	形 状 边 缘 检 测 , CANNY_THRESH=50 ,
ShapeEdgeDetection3(char*	CANNY_BLUR=12。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH, int	
CANNY_BLUR)	
void	形状边缘检测, CANNY_THRESH=50,
ShapeEdgeDetection4(char*	CANNY_BLUR=12。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH, int	
CANNY_BLUR)	
void	形状边缘检测,CANNY_THRESH2=10,
ShapeEdgeDetection5(char*	CANNY_BLUR2=2。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH2, int	
CANNY_BLUR2)	TV ID VI. th IA VIII CANDY TYPECHO 10
void	形状边缘检测,CANNY_THRESH2=10,
ShapeEdgeDetection6(char*	CANNY_BLUR2=2。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH2, int	
CANNY_BLUR2)	TZ VA 24, 44, 44, 301 CANDAY TUDDOUG 45
void	形状边缘检测,CANNY_THRESH3=45,
ShapeEdgeDetection7(char*	CANNY_blur3=10。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH3, int	
CANNY_blur3)	IX 小
void	形状边缘检测,CANNY_THRESH3=45,
ShapeEdgeDetection8(char*	CANNY_blur3=10。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH3, int	
CANNY_blur3)	

其他处理

<pre>void DES_Encrypt(char *PlainFile, char *Key, char *CipherFile)</pre>	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。
<pre>void DES_Decrypt(char *CipherFile, char *Key, char *PlainFile)</pre>	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。
void Encode(char* input,char*	文本文件压缩, input 是输入文件名,
output)	output 是输出文件名。
void Decode(char* input, char*	文本文件压缩结果解压缩, input 是
output)	输入文件名,output 是输出文件名。
void FileCompress(char *input ,	文件压缩, input 是输入文件名,
char *output)	output 是输出文件名。
void FileDecompression(char	文件压缩结果解压缩, input 是输入
*input , char *output)	文件名,output 是输出文件名。

高级算子

int*	模板匹配, input 是母本图像, Template 是样本图
TemplateMatch(char*	像,output 是结果图像文件名。channels 是图像的
input, char*	像素通道数,返回值数组中的第一个元素是最大匹配
Template, char*	分数,第二个元素是目标的 X 轴坐标值,第三个元素
output, int	万
channels, int	定日が明 1 福生が恒。主ク文持 FNG 図像。参考: channels=3, ROTATION=360 或 ROTATION=1。
ROTATION)	Chamiers-3, Korarron-300 & Korarron-1.
int*	模板匹配,input 是母本图像,Template 是样本图
	* * * * * * * * * * * * * * * * * * * *
TemplateMatch(image	像,output 是结果图像文件名。channels 是图像的
input, image	像素通道数,返回值数组中的第一个元素是最大匹配
Template, char*	分数,第二个元素是目标的 X 轴坐标值,第三个元素
output, int	是目标的 Y 轴坐标值。至少支持 PNG 图像。参考:
channels, int	channels=3, ROTATION=360 或 ROTATION=1。
ROTATION)	需引入以下头文件和结构体:
	#include "stb_image.h"
	#include "stb_image_write.h"
	typedef struct imageContainer {
	int x, y, n;
	unsigned char *data;
	} image;
	参考:
	image input, Template;
	<pre>input.data = stbi_load(inputFile, &input.x,</pre>
	&input.y, &input.n, 3); //3 表示图像有 3 个像素
	通道
	Template.data = stbi_load(templateFile,
	&Template.x, &Template.y, &Template.n, 3);
int*	模板匹配, input 是母本图像, Template 是样本图
TemplateMatch(char*	像。channels 是图像的像素通道数,返回值数组中
input, char*	的第一个元素是最大匹配分数,第二个元素是目标的
Template, int	X 轴坐标值,第三个元素是目标的 Y 轴坐标值。至少
channels, int	支持 PNG 图像。参考:channels=3,ROTATION=360 或

ROTATION)	ROTATION=1.
int*	模板匹配, input 是母本图像, Template 是样本图
TemplateMatch(image	像。返回值数组中的第一个元素是最大匹配分数,第
input, image	二个元素是目标的 X 轴坐标值,第三个元素是目标的
Template, int	Y 轴坐标值。至少支持 PNG 图像。参考: ROTATION=360
ROTATION)	或 ROTATION=1。
	需引入以下头文件和结构体:
	#include "stb_image.h"
	#include "stb_image_write.h"
	<pre>typedef struct imageContainer {</pre>
	int x, y, n;
	unsigned char *data;
	} image;
	参考:
	image input, Template;
	<pre>input.data = stbi_load(inputFile, &input.x,</pre>
	&input.y, &input.n, 3); //3 表示图像有 3 个像素
	通道
	Template.data = stbi_load(templateFile,
	&Template.x, &Template.y, &Template.n, 3);
void PGMSobel(char*	Sobel 算子,input 是输入文件名,output 是输出文
input, char*	件名。支持 P5 格式的 PGM 文件。
output, int	参考模板:
Mx[3][3], int	int $Mx[3][3] = \{\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, \}\}$
My[3][3], int max, int	0, 1}}
min)	int $My[3][3] = \{\{-1, -2, -1\}, \{0, 0, 0\}, \{1, -2, -1\}\}$
	2, 1}}
	int max = -9999
11 7010 1 11/1	int min = 9999
void PGMSobelX(char*	X 方向滤波, input 是输入文件名, output 是输出文
input, char*	件名。支持 P5 格式的 PGM 文件。
output, int	参考模板:
Mx[3][3], int	$ \begin{bmatrix} int & Mx[3][3] = \{\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, 0, 1\}\} \end{bmatrix} $
My[3][3], int max, int	$\{0, 1\}\}\$ int My[3][3] = {\{-1, -2, -1\}, \{0, 0, 0\}, \{1, \}
min)	Int my[5][5] - $\{\{-1, -2, -1\}, \{0, 0, 0\}, \{1, -2, 1\}\}$
	int max = -9999 int min = 9999
void PGMSobelY(char*	Y 方向滤波, input 是输入文件名, output 是输出文
input, char*	件名。支持 P5 格式的 PGM 文件。
output, int	
Mx[3][3], int	多有模似: int Mx[3][3] = {{-1, 0, 1}, {-2, 0, 2}, {-1,
My[3][3], int max, int	$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 $
min)	$[0, 1]$ int My[3][3] = {{-1, -2, -1}, {0, 0, 0}, {1,
11111/	Intermy $[O][O]$ (11, $[O]$, $[O]$, $[O]$, $[O]$, $[O]$

```
2, 1}}
                     int max = -9999
                     int min = 9999
                     Sobel 算子, input 是输入文件名, output 是输出文
void PGMSobel1(char*
                     件名。min 和 max 是图像归一化相关参数,如 min =
input, char*
                     1000000, max = 0; mx 和 my 分别是 Sobel 算子的 X
output, int
            min, int
max, int mx[3][3], int
                     和 Y 方向模板。支持 P2 和 P5 格式的 PGM。
                     参考模板:
my[3][3]
                     int mx[3][3] = {
                            \{-1, 0, 1\},\
                            \{-2, 0, 2\},\
                            \{-1, 0, 1\}
                        };
                         int my[3][3] = {
                            \{-1, -2, -1\},\
                            \{0, 0, 0\},\
                            {1, 2, 1}
                        };
                     X方向梯度, input 是输入文件名, output 是输出文
void
PGMSobelX1(char*
                     件名。min 和 max 是图像归一化相关参数,如 min =
                     1000000, max = 0; mx 和 my 分别是 Sobel 算子的 X
input, char*
                     和 Y 方向模板。支持 P2 和 P5 格式的 PGM。
output, int
             min, int
\max, int mx[3][3], int
                     参考模板:
                     int mx[3][3] = {
my[3][3]
                            \{-1, 0, 1\},\
                            \{-2, 0, 2\},\
                            \{-1, 0, 1\}
                        };
                         int my[3][3] = {
                            \{-1, -2, -1\},\
                            \{0, 0, 0\},\
                            \{1, 2, 1\}
                     Y方向梯度, input 是输入文件名, output 是输出文
void
                     件名。min 和 max 是图像归一化相关参数,如 min =
PGMSobelY1(char*
                     1000000, max = 0; mx 和 my 分别是 Sobel 算子的 X
input, char*
                     和 Y 方向模板。支持 P2 和 P5 格式的 PGM。
output, int
             min, int
max, int mx[3][3], int
                     参考模板:
                     int mx[3][3] = {
my[3][3]
                            \{-1, 0, 1\},\
                            \{-2, 0, 2\},\
                            \{-1, 0, 1\}
                         };
```

 $int my[3][3] = {$

	$\{-1, -2, -1\},\$
	$\{0, 0, 0\},\$
	$\{1, 2, 1\}$
	};
void PGMSobel2(char*	Sobel 算子, input 是输入文件名, output 是输出文
input, char*	件名。支持 P5 格式的 PGM 图像。XOutput 是输出的
XOutput, char*	X方向的梯度图像,Youtput 是输出的Y方向的梯度
YOutput, char*	图像,SobelOutput 是输出的整幅图像的 Sobel 算子
= '	
SobelOutput, int	计算结果, min 和 max 是图像归一化的相关参数, 如
sobel_x[3][3], int	min=100, max=0.
sobel_y[3][3], int	参考模板:
min, int max)	int sobel_x[3][3]={ $\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, \dots, 1\}$
	0, 1}};
	int sobel_y[3][3]={ $\{1, 2, 1\}, \{0, 0, 0\}, \{-1, -1\}$
	$\{2,-1\}\}$;
void Sobel(char*	Sobel 算子,input 是输入文件名,output 是输出文
input, char* output)	件名。支持 PGM 文件。
void Laplatian(char*	Laplatian 算子,input 是输入文件名,output 是输
input, char* output)	出文件名。支持 PGM 文件。
void	水平 Sobel 算子, input 是输入文件名, output 是输
HorizSobel(char*	出文件名。支持 P5 格式的 PGM 图像。
input, char* output)	
void VertSobel(char*	垂直 Sobel 算子,input 是输入文件名,output 是输
input, char* output)	出文件名。支持 P5 格式的 PGM 图像。
void PGMSobel1(char*	Sobel 算子, input 是输入文件名, output 是输出文
,	
input, char*	件名。支持 P5 格式的 PGM 图像。threshold 是目标
output, int	阈值,如 threshold=80。
threshold)	
void	Y 方向滤波, input 是输入文件名, output 是输出文
YFiltering(char*	件名。支持 P5 格式的 PGM 图像。
input, char*	参考模板:
output, int	int sobel_x[3][3] = { { 1, 0, -1},
sobe1_x[3][3], int	$\{2, 0, -2\},\$
sobe1_y[3][3])	$\{1, 0, -1\}\};$
	int sobel_y[3][3] = { { 1, 2, 1},
	$\{0, 0, 0\},$
	$\{-1, -2, -1\}\};$
void	X 方向滤波, input 是输入文件名, output 是输出文
XFiltering(char*	件名。支持 P5 格式的 PGM 图像。
input, char*	参考模板:
output, int	int sobel_x[3][3] = { { 1, 0, -1},
sobel_x[3][3], int	{ 2, 0, -2},
sobel_y[3][3])	$\{1, 0, -1\}\};$
PONCI A [9] [9])	
	int sobel_y[3][3] = $\{ \{ 1, 2, 1 \}, \}$

	$\{0, 0, 0\}, \{-1, -2, -1\}\};$
<pre>void SobelFiltering(char* input, char*</pre>	Sobel 算子, input 是输入文件名, output 是输出文件名。支持 P5 格式的 PGM 图像。 参考模板:
output, int sobel_x[3][3], int	int sobel_x[3][3] = { { 1, 0, -1}, { 2, 0, -2},
sobel_y[3][3])	$\{1, 0, -1\}\};$
	int sobel_y[3][3] = { { 1, 2, 1}, { 0, 0, 0}, $\{-1, -2, -1\}$;
void PrewittFiltering(cha r* input, char*	Prewitt 算子, input 是输入文件名, output 是输出文件名。支持 P5 格式的 PGM 图像。参考模板:
output, int	int prewitt_x[3][3] = { { 5, 5, 5},
<pre>prewitt_x[3][3], int prewitt_y[3][3])</pre>	$\{ -3, 0, -3 \}, $ $\{ -3, -3, -3 \} \};$
	int prewitt_y[3][3] = { { $5, -3, -3$ }, { $5, 0, -3$ },
void	{5, -3, -3}}; Laplace 算子, input 是输入文件名, output 是输出
LaplacianFiltering(c	文件名。支持 P5 格式的 PGM 图像。laplacian 是
har* input, char*	Laplacian 算子模板。
output, int laplacian[3][3])	参考模板: int laplacian[3][3] = { { 1, 1},
Taplacian[0][0]/	{ 1, -8, 1},
(01)	{ 1, 1, 1}};
vector <float> HarrisCornerDetectio</float>	支持 P5 和 P6 格式的 PPM 文件,从返回值数组中的 第一个元素开始,返回值以3个元素为一组,分别是
n(char* input, int	角点的 X 坐标、Y 坐标和分数, 若返回值数组名为 A,
width, int height, int	则 {A[0], A[1], A[2]} 是第一个角点的数据,
channels, int step, float	[A[3], A[4], A[5]} 是第二个角点的数据,以此类推。 input 是输入的图像文件名,width 和 height 是输
threshold, float	入图像的宽和高,channels 是输入图像的通道数,
k, float sigma)	step 默认=-1, threshold 是 Harris 检测中角点的
	得分阈值, k 是 Harris 评分函数中的 k 值, sigma 是 用 于 IxIy 阵 列 平 滑 的 sigma 值 , 参 考:
	threshold=2000, k=1, sigma=1.2。
vector(float)	支持 P5 和 P6 格式的 PPM 文件, 从返回值数组中的
HarrisCorner(char*	第一个元素开始,返回值以3个元素为一组,分别是 免点的
<pre>input, char* output, int width, int</pre>	角点的 X 坐标、 Y 坐标和分数,若返回值数组名为 A , 则 $\{A[0], A[1], A[2]\}$ 是 第 一 个 角 点 的 数 据 ,
height, int	[A[3], A[4], A[5]} 是第二个角点的数据,以此类推。
channels, float	input 是输入的图像文件名, width 和 height 是输

threshold, float 入图像的宽和高,channels 是输入图像的通道数, threshold 是 Harris 检测中角点的得分阈值, k 是 k, float sigma) Harris 评分函数中的 k 值, sigma 是用于 IxIy 阵列 平滑的 sigma 值,参考: threshold=2000, k=1, sigma=1.2. double* 模板匹配,返回值数组: 匹配框左上角顶点的 X 和 Y TemplateMatching1(Im 坐标、模板的宽和高、差异度。参考: output 是匹配 结果图像名, output_txt 是保存的匹配相关数据的 age2* input, Image2* 文本文件, threshold=0.5, isWriteImageResult=1, Template, char* output, char* color 是当图像是灰度图时的匹配框颜色, red、 output txt, double green 和 blue 是当图像是彩色图时的匹配框颜色的 红绿蓝通道值。支持 PPM 文件。 threshold, int isWriteImageResult,u 需要引入以下结构体: nsigned typedef struct Image2 char color, unsigned char red, unsigned char int width: green, unsigned char int height: blue) int channel; unsigned char* data; } Image2; 声明: Image2* readPXM(const char* name); 参考: Image2* input = readPXM(inputFileName); Image2* Template = readPXM(templatename); 模板匹配,返回值数组:匹配框左上角顶点的 X 和 Y double* TemplateMatching1(Im 坐标、模板的宽和高、差异度。参考: output txt 是 保存的匹配相关数据的文本文件, threshold=0.5, age2* input, Image2* isWriteImageResult=1, color 是当图像是灰度图时 Template, char* output txt, double 的匹配框颜色,red、green 和 blue 是当图像是彩色 图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。 threshold, unsigned 需要引入以下结构体: char color, unsigned char red, unsigned typedef struct Image2 char green, unsigned char blue) int width; int height; int channel; unsigned char* data; } Image2; 声明: Image2* readPXM(const char* name); 参考: Image2* input = readPXM(inputFileName);

Image2* Template = readPXM(templatename);

```
模板匹配,返回值数组:匹配框左上角顶点的 X 和 Y
double*
                   坐标、模板的宽和高、差异度。参考: output 是匹配
TemplateMatching2(Im
age2* input, Image2*
                   结果图像名, output txt 是保存的匹配相关数据的
                   文本文件, threshold=0.5, isWriteImageResult=1,
Template, char*
output, char*
                   color 是当图像是灰度图时的匹配框颜色, red、
output txt, double
                   green 和 blue 是当图像是彩色图时的匹配框颜色的
                   红绿蓝通道值。支持 PPM 文件。
threshold, int
                   需要引入以下结构体:
isWriteImageResult, u
nsigned
                   typedef struct Image2
              char
color, unsigned
              char
red, unsigned
              char
                      int width;
green, unsigned
              char
                      int height;
blue)
                      int channel;
                      unsigned char* data;
                   } Image2;
                   声明:
                   Image2* readPXM(const char* name);
                   参考:
                   Image2* input = readPXM(inputFileName);
                   Image2* Template = readPXM(templatename);
double*
                   模板匹配,返回值数组:匹配框左上角顶点的 X 和 Y
TemplateMatching2(Im
                   坐标、模板的宽和高、差异度。参考: output txt 是
age2* input, Image2*
                   保存的匹配相关数据的文本文件, threshold=0.5,
                   isWriteImageResult=1, color 是当图像是灰度图时
Template, char*
output txt, double
                   的匹配框颜色, red、green 和 blue 是当图像是彩色
                   图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。
threshold, unsigned
                   需要引入以下结构体:
char
     color, unsigned
char
       red, unsigned
                   typedef struct Image2
char
     green, unsigned
char blue)
                      int width:
                      int height;
                      int channel;
                      unsigned char* data;
                   } Image2;
                   声明:
                   Image2* readPXM(const char* name);
                   Image2* input = readPXM(inputFileName);
                   Image2* Template = readPXM(templatename);
                   模板匹配,返回匹配结果的图像数据。参考:
Image2*
TemplateMatching3(Im
                   output txt 是保存的匹配相关数据的文本文件,
age2* input, Image2*
                   threshold=0.5, isWriteImageResult=1, color 是当
                   图像是灰度图时的匹配框颜色, red、green 和 blue
Template,
             char*
```

output_txt,

double

是当图像是彩色图时的匹配框颜色的红绿蓝通道值。

```
threshold,
                     需要引入以下结构体:
                int
isWriteImageResult,
                     typedef struct Image2
unsigned char color,
unsigned char red,
                        int width:
unsigned char green,
                        int height;
unsigned char blue)
                        int channel;
                        unsigned char* data;
                     } Image2;
                     声明:
                     Image2* readPXM(const char* name);
                     参考:
                     Image2* input = readPXM(inputFileName);
                     Image2* Template = readPXM(templatename);
                     图像匹配,返回匹配结果的图像数据。参考:
Image2*
TemplateMatching4(Im
                     output txt 是保存的匹配相关数据的文本文件,
age2* input, Image2*
                     threshold=0.5, isWriteImageResult=1, color 是当
                     图像是灰度图时的匹配框颜色, red、green 和 blue
Template,
              char*
                     是当图像是彩色图时的匹配框颜色的红绿蓝通道值。
output txt,
             double
                     需要引入以下结构体:
threshold,
                int
isWriteImageResult,
                     typedef struct Image2
unsigned char color,
unsigned char red,
                        int width;
unsigned char green,
                        int height;
unsigned char blue)
                        int channel;
                        unsigned char* data;
                     }Image2:
                     声明:
                     Image2* readPXM(const char* name);
                     参考:
                     Image2* input = readPXM(inputFileName);
                     Image2* Template = readPXM(templatename);
                     模板匹配,返回匹配框的中心点坐标、旋转角度和缩
double*
TemplateMatching (RGB
                    放比例。参考: c=0.5, threshold=0.9。支持 PPM 文
PACKED IMAGE*
                     件。
                     声明:
input,
RGB PACKED IMAGE*
                    #ifndef P
                    #if defined( STDC ) | defined( cplusplus)
Template, char*
                    #define P(protos) protos
output, unsigned char
red, unsigned
               char
                    #else
green, unsigned
                    #define __P(protos) ()
               char
blue, double c, double
                    #endif
threshold)
                    #endif
                    RGB PACKED IMAGE
                                             *readRGBPackedImage
                      P(( char* ));
```

```
需引入以下结构体:
                      typedef struct rgb packed pixel {
                         BYTE r;
                         BYTE g;
                         BYTE b;
                      } RGB PACKED PIXEL;
                     typedef struct rgb packed image {
                         int cols;
                         int rows;
                         RGB_PACKED_PIXEL **p;
                         RGB PACKED PIXEL *data p;
                     } RGB_PACKED_IMAGE;
                      参考:
                     RGB PACKED IMAGE* Template =
                     readRGBPackedImage(templatename);
                     RGB PACKED IMAGE*
                                                 input
                     readRGBPackedImage(inputFileName);
double*
                     模板匹配,返回匹配框的中心点坐标、旋转角度和缩
                     放比例。参考: c=0.5, threshold=0.9。支持 PPM 文
TemplateMatching1 (RG
B PACKED IMAGE*
                     件。
                      声明:
input,
RGB PACKED IMAGE*
                     #ifndef P
Template, unsigned
                     #if defined( STDC ) | defined( cplusplus)
                     #define P(protos) protos
char
        red, unsigned
                     #else
char
      green, unsigned
         blue, double
                     #define __P(protos) ()
char
c, double threshold)
                     #endif
                     #endif
                     RGB_PACKED_IMAGE
                                         *readRGBPackedImage
                       P(( char* ));
                      需引入以下结构体:
                      typedef struct rgb packed pixel {
                         BYTE r:
                         BYTE g;
                         BYTE b:
                     } RGB_PACKED_PIXEL;
                     typedef struct rgb packed image {
                         int cols;
                         int rows;
                         RGB_PACKED_PIXEL **p;
                         RGB PACKED PIXEL *data p;
                     } RGB PACKED IMAGE;
                      参考:
                     RGB PACKED IMAGE*
                                               Template
```

```
readRGBPackedImage(templatename);
                     RGB PACKED IMAGE*
                                                 input
                     readRGBPackedImage(inputFileName);
                     模板匹配, 返回匹配框的中心点坐标、旋转角度和缩
double*
TemplateMatching2(RG
                     放比例。参考: c=0.5, threshold=0.9。 支持 PPM 文
                     件。
B PACKED IMAGE*
                     声明:
input,
RGB PACKED IMAGE*
                     #ifndef P
                     #if defined(_STDC__) || defined(_cplusplus)
Template, unsigned
        red, unsigned
                     #define __P(protos) protos
char
char
      green, unsigned
                     #else
                     #define __P(protos) ()
         blue, double
char
c, double threshold)
                     #endif
                     #endif
                     RGB PACKED IMAGE
                                         *readRGBPackedImage
                      P(( char* ));
                     需引入以下结构体:
                     typedef struct rgb packed pixel {
                         BYTE r;
                         BYTE g;
                         BYTE b:
                     } RGB PACKED PIXEL;
                     typedef struct rgb packed image {
                         int cols;
                         int rows;
                         RGB_PACKED_PIXEL **p;
                         RGB PACKED PIXEL *data p;
                     } RGB PACKED_IMAGE;
                     参考:
                     RGB PACKED_IMAGE*
                                               Template
                     readRGBPackedImage(templatename);
                     RGB PACKED IMAGE*
                                                 input
                     readRGBPackedImage(inputFileName);
                     模板匹配,返回匹配结果的图像数据。参考: c=0.5,
RGB PACKED IMAGE*
TemplateMatching (RGB
                     threshold=0.9。支持 PPM 文件。
PACKED IMAGE*
                     声明:
                     #ifndef P
input,
                     #if defined( STDC ) | defined( cplusplus)
RGB PACKED IMAGE*
Template, unsigned
                     #define P(protos) protos
char red.
                     #else
           unsigned
char green, unsigned
                     #define P(protos) ()
char blue, double c,
                     #endif
double threshold)
                     #endif
                     RGB PACKED IMAGE
                                               *readRGBPackedImage
```

```
P(( char* ));
                  需引入以下结构体:
                  typedef struct rgb packed pixel {
                      BYTE r;
                      BYTE g;
                      BYTE b;
                  } RGB PACKED PIXEL;
                  typedef struct rgb packed image {
                      int cols;
                      int rows;
                      RGB PACKED PIXEL **p;
                      RGB PACKED PIXEL *data p;
                  } RGB PACKED IMAGE;
                  参考:
                  RGB_PACKED_IMAGE* Template =
                  readRGBPackedImage(templatename);
                  RGB PACKED IMAGE*
                  readRGBPackedImage(inputFileName);
                  特征检测,返回特征点数量。支持 PPM 文件。
unsigned
              int
FeatureDetection(cha
        input, char*
output, unsigned char
red, unsigned
              char
green, unsigned
              char
blue)
vector<int>
                  特征匹配,返回值中的数据格式为:第1个值是特征
                  对的序号,从0开始,第2个值和第3个值分别是是
FeatureMatching (char
       input1, char*
                  图像1的其中一个特征点的横坐标和纵坐标,第4个
                  值和第 5 个值分别是是图像 2 的其中一个特征点的
input2, char*
                  横坐标和纵坐标并与第 2 个值和第 3 个值所在的图
output, unsigned char
                  像1的特征点相对应,这5个值构成一组;同理,第
red, unsigned
              char
             char
                  6个值就是下一个特征对的序号,即为1,后面的值
green, unsigned
blue)
                  则以此类推。格式如:特征对:%d,图像1(%d,%d)->
                  图像 2: (%d, %d)。 支持 PPM 文件。
                  从两幅输入图像中检测特征点,然后使用蛮力方法匹
unsigned
              int
                  配两幅图像的特征。input 是输入图像, output 是生
FeatureMatching1 (cha
                  成的特征点图像,bExtractDescriptor=true。返回
r*
       input1, char*
                  匹配的特征点数量。支持 PGM 文件。
input2, char*
output, unsigned char
red, unsigned
              char
green, unsigned
              char
blue, bool
bExtractDescriptor)
                  返回值中的数据格式为: 第1个值是特征对的序号,
vector<int>
```

FeatureMatching2(cha
r* input1, char*
input2, char*
output, unsigned char
red, unsigned char
green, unsigned char
blue, bool
bExtractDescriptor)

从 0 开始,第 2 个值和第 3 个值分别是是图像 1 的 其中一个特征点的横坐标和纵坐标,第 4 个值和第 5 个值分别是是图像 2 的其中一个特征点的横坐标和 纵坐标并与第 2 个值和第 3 个值所在的图像 1 的特 征点相对应,这 5 个值构成一组;同理,第 6 个值就 是下一个特征对的序号,即为 1,后面的值则以此类 推。格式如:特征对:%d,图像 1 (%d,%d) ->图像 2: (%d,%d)。input 是输入图像,output 是生成的特征 点图像,bExtractDescriptor=true。支持 PGM 文件。 返回特征点数量。input 是输入图像,output 是生成

unsigned int
FeatureExtraction1(c
har* input, char*
output, unsigned char
red, unsigned char
green, unsigned char
blue, bool
bExtractDescriptor)

返回特征点数量。input 是输入图像, output 是生成的特征点图像, bExtractDescriptor=true。支持 PGM 文件。

std::list<ezsift::Si
ftKeypoint>
FeatureExtraction2(c
har* input, char*
output, unsigned char
red, unsigned char
green, unsigned char
blue, bool
bExtractDescriptor)

返回特征点列表。input 是输入图像, output 是生成的特征点图像, bExtractDescriptor=true。支持 PGM 文件。 需引入以下命名空间:

namespace ezsift {
#define DEGREE (128) //SIFT 关键点: 128 维
struct SiftKeypoint {
 int octave; // octave 数量

// layer 数量 int layer; float rlayer; // layer 实际数量 // 归一化的 row 坐标 float r: // 归一化的 col 坐标 float c; float scale; // 归一化的 scale // row 坐标(layer) float ri; // column 坐标(layer) float ci; float layer_scale; // scale(layer) float ori; // 方向(degrees) float mag; // 模值 float descriptors[DEGREE]; //描述符

vector<int>
DefectLocation(PGMDa
ta*
Template, PGMData*
Sample, int floor, int

size, int a, int b, int

};

查找缺陷位置,返回材料缺陷的位置,以每4个元素为一组。Template是模板图像,Sample是样本图像,g是缺陷的有效界限的X轴长度,h是缺陷的有效界限的Y轴长度,参考:floor=80,size=10,a=64,b=64,c=16,d=16,e=2,f=4,g=65,h=65,FULL=0,EMPTY=255,report=true。

```
引入以下结构体:
c, int
       d, int
              e, int
                    typedef struct PGMData {
f, int
       g, int
              h, int
FULL, int
         EMPTY, bool
                        int row;
report)
                        int col;
                        int max_gray;
                        int **matrix;
                    } PGMData;
                    若模板文件名为 Template, 样本文件名为 Sample,
                    使用以下代码获得合适的输入数据:
                    首先声明 readPGM 函数:
                    PGMData*
                              readPGM(const
                                              char
                                                    *file name,
                    PGMData *data):
                    然后执行以下代码:
                    PGMData*
                                          model
                                                              =
                     (PGMData*) malloc(sizeof(PGMData));
                    readPGM(Template, model);
                    PGMData*
                                           data
                     (PGMData*) malloc(sizeof(PGMData));
                    readPGM(Sample, data);
                    之后将 model 和 data 传入相应的函数。
                     支持 P5 格式的 PGM 文件。
                    缺陷尺寸,返回缺陷尺寸,以4个为一组。Template
vector<int>
DefectSize(PGMData*
                    是模板图像, Sample 是样本图像, g 是缺陷的有效界
Template, PGMData*
                    限的 X 轴长度, h 是缺陷的有效界限的 Y 轴长度,参
Sample, int floor, int
                    考: floor=80, size=10, a=64, b=64, c=16, d=16,
                    e=2, f=4, g=65, h=65, FULL=0, EMPTY=255,
size, int a, int b, int
c, int
       d, int
                    report=true.
              e, int
f, int
       g, int
              h, int
                    引入以下结构体:
FULL, int
         EMPTY, boo1
                    typedef struct _PGMData {
report)
                        int row;
                        int col;
                        int max gray;
                        int **matrix;
                    } PGMData;
                    若模板文件名为 Template, 样本文件名为 Sample,
                    使用以下代码获得合适的输入数据:
                     首先声明 readPGM 函数:
                    PGMData*
                              readPGM(const
                                                    *file name,
                                              char
                    PGMData *data);
                    然后执行以下代码:
                    PGMData*
                                          model
                     (PGMData*) malloc(sizeof(PGMData));
                    readPGM(Template, model);
                    PGMData*
                                           data
```

	(PGMData*)malloc(sizeof(PGMData));
	readPGM(Sample, data);
	之后将 model 和 data 传入相应的函数。
	支持 P5 格式的 PGM 文件。
vector <int></int>	样品好坏的数量,返回结果中,第一个元素是合格的
GoodBadQuantity(PGMD	圆圈数量,第二个元素是缺陷的圆圈数量。Template
ata*	是模板图像,Sample 是样本图像,g 是缺陷的有效界
Template, PGMData*	限的 X 轴长度, h 是缺陷的有效界限的 Y 轴长度,参
Sample, int floor, int	考: floor=80, size=10, a=64, b=64, c=16, d=16,
size, int a, int b, int	e=2, f=4, g=65, h=65, FULL=0,EMPTY=255,
c, int d, int e, int	report=true。
f, int g, int h, int	引入以下结构体:
FULL, int EMPTY, bool	typedef struct _PGMData {
report)	int row;
	int col;
	int max_gray;
	int **matrix;
	}PGMData;
	若模板文件名为 Template,样本文件名为 Sample,
	使用以下代码获得合适的输入数据:
	首先声明 readPGM 函数:
	PGMData* readPGM(const char *file_name,
	PGMData *data);
	然后执行以下代码:
	PGMData* mode1 =
	(PGMData*)malloc(sizeof(PGMData));
	readPGM(Template, model);
	PGMData* data =
	(PGMData*)malloc(sizeof(PGMData));
	readPGM(Sample, data);
	之后将 model 和 data 传入相应的函数。
	支持 P5 格式的 PGM 文件。
void	Sobel 算子,input 是输入文件名,output 是输出文
RAWSobelEdge(char*	件名。ROWS 是图像的行,COLS 是图像的列,M 是滤
input, char*	波相关参数,如 M=1。支持 RAW 图像。
output, int ROWS, int	参考模板:
COLS, int M, float	float sobelX[3][3] = $\{\{-1,0,1\},$
sobelX[3][3], float	$\{-2, 0, 2\},$
sobe1Y[3][3])	$\{-1, 0, 1\}\};$
	float sobelY[3][3] = $\{\{-1, -2, -1\},$
	110at soberr[5][5] - $\{\{-1, -2, -1\}, \{0, 0, 0\}, \}$
	$\{0,0,0\},\ \{1,2,1\}\};$
void	边缘检测, input 是输入文件名, output 是输出文件
voiu	应纳亚州,Imput 定制八人什石,Output 定制出人件

RAWPlaceHolder(char*	名。ROWS 是图像的行, COLS 是图像的列, M 是滤波
input, char*	相关参数,如 M=1。支持 RAW 图像。
output, int ROWS, int	参考模板:
COLS, int M, float	float $mask[3][3] = \{\{-1, -2, -1\},\$
mask[3][3])	$\{0,0,0\},\{1,2,1\}\};$
void	拉普拉斯锐化滤波器, input 是输入文件名, output
RAWLaplacialSharpeni	是输出文件名。ROWS 是图像的行大小,COLS 是图像
ngFilter(char*	的列大小, M 和 w 是滤波相关参数, 如 M=1, w=1;
input, char*	mask 是滤波器模板。支持 RAW 图像。
output, int ROWS, int	参考模板:
COLS, int M, float	float $mask[3][3] = \{\{0, 1, 0\},\$
w,float mask[3][3])	{1, -4, 1},
	$\{0, 1, 0\}\};$
void	拉普拉斯算子增强, input1 是输入的 RAW 图像文件
RawLaplacianEnhancem	名,output1 是输出的 RAW 图像文件名,width 是输
ent (char*	入图像的宽,height 是输入图像的高。支持 RAW 图
input1, char*	像。
output1, int	
width, int height)	
struct	圆检测,返回找到的圆的位置和大小等相关信息。支
hough_param_circle*	持 RAW 文件。
CircleDetection(char	需引入以下结构体:
* input, int	struct hough_param_circle {
width, int height)	int a;
	int b;
	int radius;
	int resolution;
	int thresh;
	struct point *points;
	int points_size;
	};
void	圆样变形,支持 RAW 文件。
ImageWarpEllipticalG	
rid(string	
input, string	
output, int	
height, int width, int	
NumberChannels)	
int* FindLine(char*	直线检测,返回直线的 theta 和 rho,支持 RAW 图
input, char*	像。
output, int width, int	
height)	
int*	圆检测,返回圆心的坐标和圆的半径,支持 RAW 图

input, char*	
_ ·	
output, int width, int	
height, float sigma,	
int tmin, int tmax)	Z- E-1A/EU - 1 - 1 - 1 - 1 - 1 - 1 - 0 - 0 - 0 - 0
void	角点检测,threshold=10000,k=0.06,sigma=1.0,
CornerDetection(char	width=640, height=480, channels=1。支持 PNM 图
* input, char*	像。
output, float	
threshold, float k,	
float sigma, int	
width, int height,	
int channels)	
vector <keypoint></keypoint>	角点检测,返回角点数据。threshold=10000,k=0.06,
CornerDetection1(cha	sigma=1.0, width=640, height=480, channels=1.
r* input, char*	支持 PNM 图像。
output, float	需引入以下结构体
threshold, float k,	typedef struct {
float sigma, int	float x;
width, int height,	float y;
int channels)	float score;
,	} Keypoint;
vector (Corner:: Keypo	角点检测,返回角点数据。threshold=2000, k=1,
int>	sigma=1.2。支持 PNM 图像。
CornerDetection(char	需引入以下命名空间:
* input, int	namespace Corner {
width, int height, int	struct Keypoint {
channels, float	float x;
threshold, float k,	float y;
float sigma)	float score;
11000 a 18mon	};
void Structure(char*	 特征归一化统计,参考: sigma=2。支持多种图像格
input, char*	式。
output, float sigma)	
void	角点检测,参考: sigma=2, method=0。支持多种图
Cornerness (char*	像格式。
input, char*	N IH ~ V°
output, float	
sigma, int method)	
void Corners (char*	角点检测,参考: sigma=2, thresh=0.4, window=5,
input, char*	nms=3, corner_method=0。支持多种图像格式。
	Innis 0, Colliel_method-0。又可多种图像作人。
output, float sigma,	
float thresh, int	
window, int nms, int	

corner method)

void FindMatch(char* input1, char* input2, char* output, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int corner method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, nms5, int int corner method5, float sigma6, int corner method6, float thresh6, int window6, int nms6, float inlier thresh6, int iters6, int cutoff6, float acoeff6)

特征匹配,参考: thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner_method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3, corner_method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner_method2=0, sigma5=2, thresh5=0.4, window5=7, nms5=3, corner_method5=0, sigma6=2, corner_method6=0, thresh6=0.3, window6=7, nms6=3, inlier_thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5。支持多种图像格式。

vector \ Descriptor \>
HarrisCorner(char*
input, char*
output, float sigmal,
float threshl, int
windowl, int nmsl,
int corner_methodl)

角点检测,返回检测结果。参考: sigmal=2, thresh1=0.4 , window1=5 , nms1=3 , corner_method1=0。支持多种图像格式。需引入以下结构体: struct Point { double x, y;

Point(): x(0), y(0) {}
Point(double x, double y): x(x), y(y) {}

```
struct Descriptor {
                        Point p;
                        vector<float> data;
                        Descriptor() {}
                        Descriptor(const Point& p) : p(p) {}
vector < Match >
                      描述匹配项,返回描述结果。参考: sigma1=2,
MatchDescriptors (cha
                      thresh1=0.4
                                        window1=5,
                                                         nms1=3
        input1, char*
                      corner method1=0, sigma2=2, thresh2=0.4,
input2, char*
                      window2=5, nms2=3, corner method2=0。支持多种
                      图像格式。
output, float sigmal,
float threshl,
                      需引入以下结构体:
                 int
window1,
          int nms1,
                      struct Point {
int
                        double x, y;
corner method1, float
                        Point(): x(0), y(0) {}
sigma2,
thresh2,
                 int
                        Point (double x, double y) : x(x), y(y) {}
window2,
          int nms2,
                      struct Descriptor {
int corner method2)
                        Point p;
                        vector<float> data;
                        Descriptor() {}
                        Descriptor(const Point& p) : p(p) {}
                      };
                      struct Match {
                        const Descriptor* a=nullptr;
                        const Descriptor* b=nullptr;
                        float distance=0.f;
                        Match() {}
                        Match (const Descriptor* a, const Descriptor*
                      b, float dist=0. f): a(a), b(b), distance(dist)
                      {}
                        bool operator (const Match& other) { return
                      distance<other.distance; }</pre>
                      绘制角点,参考: thresh3=5, k=10000, cutoff=50,
void
DrawInliers(char*
                      thresh4=5, sigma=2, thresh=0.4, window=5, nms=3,
input1, char*
                      corner method=0, sigma1=2, thresh1=0.4,
input2, char*
                      window1=5, nms1=3, corner_method1=0, sigma2=2,
output, float
                      thresh2=0.4
                                        window2=5
                                                         nms2=3
```

thresh3, int k, int cutoff, float thresh4, float sigma, float thresh. int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int corner method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, nms5, int int corner method5, float sigma6, int corner method6, float thresh6, int window6. int nms6. float inlier_thresh6, iters6, int cutoff6, float acoeff6)

corner_method2=0 , sigma5=2 , thresh5=0.4 , window5=7, nms5=3 , corner_method5=0 , sigma6=2 , corner_method6=0 , thresh6=0.3 , window6=7 , nms6=3 , inlier_thresh6=5 , iters6=1000 , cutoff6=50 , acoeff6=0.5 。 支持多种图像格式。

void PanoramaImage(char* input1, char* input2, char* output, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float int thresh1, window1, nms1, int int

制造全景图像,参考: thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner method=0, sigma1=2, thresh1=0.4 , window1=5 nms1=3corner_method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner method2=0, sigma5=2, window5=7 thresh5=0.4nms5=3corner method5=0, sigma6=2, corner method6=0, thresh6=0.3 window6=7 nms6=3inlier thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5。支持多种图像格式。

corner method1, float float sigma2, int thresh2, window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, int nms5, int corner method5, float sigma6, corner method6, float thresh6, int window6, int nms6, float. inlier thresh6, iters6, int cutoff6, float acoeff6)

角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3, corner method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner method2=0, sigma5=2, thresh5=0.4window5=7 nms5=3corner method5=0, sigma6=2, corner method6=0, thresh6=0.3window6=7 inlier_thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5, sigma7=2, corner method7=0, thresh7=0.3 window7=7 inlier thresh7=5, iters7=1000, cutoff7=50, acoeff7=0.5。支持多种图像格式。

void Cylindrical (char* input1, char* input2, char* output, float f2, float f1, float thresh3, int k, int cutoff, float thresh4, float sigma, thresh, float int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int corner_method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, int nms5,

int corner method5, float sigma6, int corner method6, float thresh6, int window6, int nms6, float inlier thresh6, int iters6, int cutoff6, float acoeff6, float sigma7, int corner_method7, float thresh7, int window7, int nms7, float inlier_thresh7, iters7, int cutoff7, float acoeff7)

角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3, corner method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner method2=0, sigma5=2, thresh5=0.4window5=7, nms5=3corner_method5=0, sigma6=2, corner_method6=0, thresh6=0.3 window6=7, nms6=3inlier thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5 , sigma7=2 , corner_method7=0 , thresh7=0.3, window7=7, nms7=3inlier thresh7=5, iters7=1000, cutoff7=50, acoeff7=0.5。支持多种图像格式。

void Spherical (char* input1, char* input2, char* output, float f1, float f2, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner_method, float sigmal, float thresh1, int window1, int nms1, int corner_method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, int nms5, int corner method5, float

```
sigma6,
                 int
corner method6,
      thresh6,
float
                 int
window6.
          int nms6,
float
inlier thresh6,
                 int
iters6, int cutoff6,
float
       acoeff6, float
sigma7,
                 int
corner_method7,
float thresh7,
                 int
window7,
          int nms7,
float
inlier thresh7,
iters7, int cutoff7,
float acoeff7)
void
                     特征匹配,支持 PGM 图像。
FeatureMatching (char
        input1, char*
input2, char* output)
                     特征匹配,返回匹配的点对,支持 BMP 图像。
int
FeatureMatching (char
        input1, char*
input2, char*
output, char*
outputCouple)
                      圆检测,返回检测结果。size1=5, size2=5, size3=7。
std::vector
<CentersPoint>
                      支持 BMP 图像。
                      需引入以下结构体:
FindCircles(char*
input, char*
                     struct Point {
output, int size1, int
                         Point (int x = 0, int y = 0) { this->x = x;
size2, int size3)
                      this->y = y; }
                         int x;
                         int y;
                     };
                      struct CentersPoint {
                         CentersPoint (Point
                                            point,
                                                      int
                                                           radius)
                      { this->point = point; this->radius = radius;
                      count = 1;
                         Point point;
                         int count;
                         int radius;
                      };
                     Canny 算子, 至少支持 JPG 图像, input 是输入文件
void
         Canny (char*
```

input, char*	名, output 是输出文件名,参考: lowThreshold=50,
output, int	highThreshold=150。
lowThreshold, int	
highThreshold)	
void KMeans1(char*	K-Means 聚类,input 是输入文件名,output 是输出
input, char*	文件名。输入图像最好宽高相同,c的最大值是图像
output, int c, int k)	的宽和高中较小的那个参数,如宽=500,高为600,
	则 c 最大可取 500; k 是聚类的种类数目。支持 BMP
	图像。
void KMeans(string	K-Means 聚类,input 是输入文件名,Clusters 是聚
input, unsigned int	类的种类数目, output 是输出文件名。支持 BMP 图
Clusters, char*	像。
output)	
void LSBRead(char*	LSB 隐写文件的读出, input 是输入文件名, output
input, char*	是输出文件名。参考: color1=255, color2=0。支持
output, int width, int	24 位 BMP 图像。
height, unsigned char	p. 2 p. 10
color1, unsigned char	
color2)	
void LSBWrite(char*	LSB 隐写, input1 是用于容纳隐藏图像的图像,
input1, char*	input2 是要隐藏的黑白图像,参考: threshold1、
input2, char*	threshold2 和 threshold3 都 等 于 128 ,
	color1=(unsigned char) $0b00000001$,
output, int width, int height, unsigned char	color1=(unsigned char) 0b11111110。支持24位
thresholdl, unsigned	BMP 图像。
char	DIMI 国家。
threshold2, unsigned	
char	
threshold3, unsigned	
char colorl, unsigned	
char color2)	
void	Roberts 算子,input 是输入数据,output 是输出数
Roberts (unsigned	据。
char**	∜⊢ ∨
input, unsigned	
char** output)	
void	Roberts 算子,input 是输入数据,output 是输出数
Roberts (BMPMat**	据。
input, BMPMat**	▼ -
output)	
void	STL 切片,input 是输入的 STL 文件,output 是输出
STLSection(char*	的切片文件前缀名,sliceAmount 是切片量,如:
input, char*	sliceAmount=50, resolution 是分辨率,如:
output, int	resolution=260, c 是执行的相关参数,如: c=5。
output, Int	10001dt10f1 200, 0 人以(11月/1日/12/9X, 为日, 0 0。

1	
sliceAmount, int	
resolution, int c)	
void SURF(char*	SURF 算子, input1 和 input2 是输入文件名, output
input1, char*	是输出文件名。支持 BMP 图像。
input2, char* output)	
void	二进制法 Sobel 算子,input1 和 input2 是输入文件
SobelBinary(char*	名,output 是输出文件名。支持 BMP 图像。
input, char*	参考:
output, char	$char filterH[9] = \{-1, 0, 1,$
filterH[9], char	-2, 0, 2,
filterV[9])	-1, 0, 1};
	$char filterV[9] = \{1, 2, 1,$
	0, 0, 0,
	$-1, -2, -1\};$
void	Sobel 算子,耗时较长, input 是输入文件名, output
SobelOperator(char*	是输出文件名。支持 24 位 BMP 图像。
input, char* output)	
SobelImage**	返回处理后的各像素点的坐标和对应的像素值,若是
SobelOperator(char*	边缘点则对应白色,否则对应黑色。支持 BMP 图像。
input)	需引入以下结构体:
	typedef struct {
	int x;
	int y;
	unsigned char red;
	unsigned char green;
	unsigned char blue;
	}SobelImage;
void Sobel2(char*	Sobel 算子, input 是输入文件名, output 是输出文
input, char*	件名。支持 BMP 图像。
output, char	参考:
filterH[9], char	char filterH[9] = {-1, 0, 1,
filterV[9])	-2, 0, 2,
,	-1, 0, 1;
	char filterV[9] = {1, 2, 1,
	0, 0, 0,
	$-1, -2, -1\};$
void	边缘检测, input 是输入文件名, output 是输出文件
EdgeDetection(char*	名。支持4位BMP图像。
input, char* output)	
void	边缘检测, input 是输入文件名, output 是输出文件
EdgeDetection1(char*	名。支持 8 位 BMP 图像。
input, char*	参考模板:
output, short	short sharpen[3][3] = $\{\{1, 1, 1\},$
sharpen[3][3])	{1, -8, 1},
1 /	. , , , , ,

	{1, 1, 1}};
void	边缘检测, input 是输入文件名, output 是输出文件
EdgeDetection2(char*	名。a 是用于设置图像像素的相关参数,如 a=3。支
input, char*	持 24 位 BMP 图像。
output, int a)	10 21 12 pm
void	 边缘检测,input 是输入文件名,output 是输出文件
EdgeDetection3(char*	名。a 是用于设置图像像素的相关参数,如 a=3。支
input, char*	持 24 位 BMP 图像。
output, int a)	14 9.1 Tr. Dim Ed 19/40
void	 边缘检测,input 是输入文件名,output 是输出文件
EdgeDetection4(char*	名。a 是用于设置图像像素的相关参数,如 a=3。支
input, char*	持 24 位 BMP 图像。
output, int a)	14 51 17 pm
void Roberts(char*	Roberts 边缘检测。支持 BMP 图像。
input, char* output)	TOOLEON CONTENTS AND DIE DIE
void Prewitt(char*	 Prewitt 边缘检测。支持 BMP 图像。
input, char* output)	THOUTHOU ZUN ELMO
void Sobel(char*	Sobel 边缘检测。支持 BMP 图像。
input, char* output)	SOSSET CONTENTS X14 DIM PINE
void Laplace(char*	Laplace 边缘检测。支持 BMP 图像。
input, char* output)	Dapidoo Cox EM 211 Din EM
void	二维码生成, filename 是生成的二维码图像文件名,
QRCodeGeneration(cha	inputString 是二维码包含的信息。
r *filename, char*	
inputString)	
void	二维码编码,输入文本文件,输出二维码的 PBM 图
QRCodeEncode(char*	像。
input, char* output)	
void	二维码解码,输入为二维码的 PBM 图像,输出为文本
QRCodeDecode(char*	文件。
input, char* output)	
void	模板匹配, a=0.5, b=0, a1=0.5, b1=0.5, c=0.2,
TemplateMatching(cha	ps 是相似度,如 ps=0.5。支持 24 位 BMP 图像。
r* input, char*	
Template, char*	
output, unsigned int	
b, double ps, double	
a, double al, double	
b1, double c)	
SearchResult	模板匹配,返回目标位置的左上角坐标和相似值。
TemplateMatching(cha	支持 PNG 图像。
r* input, char*	需引入以下结构体:
Template)	struct SearchResult {
	int x, y;

```
double value;
SearchResult
                     模板匹配,返回目标位置的左上角坐标和相似值。
TemplateMatching(uin
                     需引入以下头文件:
                     #define STB_IMAGE_IMPLEMENTATION
t8 t* input, uint8 t*
                     #include "stb image.h"
Template, int
imgWidth, int
                     #include <cstdint>
imgHeight, int
                     #include <complex>
                     #include <vector>
imgBpp, int
patWidth, int
                     参考例程:
patHeight, int
                     int imgWidth, imgHeight, imgBpp;
patBpp)
                     int patWidth, patHeight, patBpp;
                                         =
                                             stbi load(inputfile,
                     uint8 t*
                                input
                     &imgWidth, &imgHeight, &imgBpp, 3);
                     uint8 t* Template = stbi load(Templatefile,
                     &patWidth, &patHeight, &patBpp, 3);
                     模板匹配, ps 是相似度,
void
TemplateMatching1(ch
                     colors=(image colors) \{255, 0, 0\}.
                     支持 24 位 BMP 图像。
ar*
         input, char*
                     需引入以下结构体:
Template, char*
output, float
                     typedef struct {
ps, image_colors
                         unsigned char R, G, B;
colors)
                     } image colors;
                     模板匹配, ps 是相似度, ps=0.8,
void
                     colors=(image colors) {255, 0, 0}。支持24位BMP
TemplateMatching1(im
age
         input, image
                     图像。
                     需引入以下结构体和头文件以及声明:
Template, char*
                     #include <stdint.h>
output, float
ps, image colors
                     typedef struct {
colors)
                         uint16 t signature;
                         uint32_t size;
                         uint16 t reserved1;
                         uint16_t reserved2;
                         uint32 t offset;
                         uint32 t bitmapinfo;
                         int32 t width;
                         int32_t height;
                         uint16_t no_planes;
                         uint16 t no bits pixel;
                         uint32_t compression;
                         uint32 t size padding;
                         int32_t x_pixe1_meter;
                         int32_t y_pixe1_meter;
                         uint32_t no_colors;
```

```
uint32_t no_imp_colors;
                     } image header;
                     typedef struct {
                         unsigned char R, G, B;
                     } image_colors;
                     typedef struct {
                         image colors *pixels;
                                                      // Imaginea
                     in forma liniarizata
                                                     // Header-ul
                         image header header;
                     imaginii
                         int32_t padding;
                     } image;
                     bool
                            grayscale image(char*
                                                   path to image,
                     char* path to grey);
                     image load_image(char* path_to_image);
                     参考例程:
                     char *path_to_grey="gray.bmp";
                     grayscale image (inputfile, path to grey);
                     image input, Template;
                     input = load_image(path_to_grey);
                     Template = load_image(Templatefile);
                     模板匹配, ps 是相似度, ps=0.8,
window
TemplateMatching3(ch
                     colors=(image colors) {255, 0, 0}。支持24位BMP
        input, char*
                     图像。
Template, float
                     需引入以下头文件和结构体:
ps, image colors
                     #include <stdint.h>
colors)
                     typedef struct {
                         unsigned char R, G, B;
                     } image_colors;
                     typedef struct {
                         uint32 t x, y;
                         image colors colors;
                         double ps;
                     \} x0y:
                     typedef struct {
                         uint32_t width, height, matches;
                         x0y *pos;
                     } window;
                     模板匹配, ps 是相似度, ps=0.8,
window
TemplateMatching3(im
                     colors=(image colors) {255, 0, 0}。支持24位BMP
        input, image
                     图像。
age
                     需引入以下结构体和头文件以及声明:
Template, float
ps, image_colors
                     #include <stdint.h>
colors)
                     typedef struct {
```

```
uint16_t signature;
                         uint32 t size;
                          uint16_t reserved1;
                         uint16 t reserved2;
                         uint32_t offset;
                         uint32 t bitmapinfo;
                          int32_t width;
                          int32 t height;
                         uint16 t no planes;
                          uint16_t no_bits_pixel;
                         uint32 t compression;
                          uint32_t size_padding;
                          int32 t x pixel meter;
                          int32_t y_pixel_meter;
                         uint32_t no_colors;
                         uint32_t no_imp_colors;
                      } image header;
                      typedef struct {
                         unsigned char R, G, B;
                      } image colors;
                      typedef struct {
                          image_colors *pixels;
                                                       // Imaginea
                      in forma liniarizata
                          image header header;
                                                      // Header-ul
                      imaginii
                          int32_t padding;
                     } image;
                     bool
                            grayscale_image(char*
                                                    path to image,
                      char* path_to_grey);
                      image load image(char* path to image);
                      参考例程:
                      char *path to grey="gray.bmp";
                      grayscale image (inputfile, path to grey);
                      image input, Template;
                      input = load image(path to grey);
                      Template = load_image(Templatefile);
                      模板匹配, ps 是相似度, ps=0.8,
window*
TemplateMatching5(ch
                      colors=(image colors) {255, 0, 0}。支持 24 位 BMP
                      图像。
ar*
         input, char*
Template, float
ps, image colors
colors)
                     模板匹配, ps 是相似度, ps=0.8,
window*
                     colors=(image_colors) {255, 0, 0}。支持 24 位 BMP
TemplateMatching5(im
```

```
图像。
         input, image
age
                      需引入以下结构体和头文件以及声明:
Template, float
ps, image colors
                      #include <stdint.h>
colors)
                      typedef struct {
                          uint16_t signature;
                          uint32 t size;
                          uint16 t reserved1;
                          uint16 t reserved2;
                          uint32 t offset;
                          uint32_t bitmapinfo;
                          int32 t width;
                          int32_t height;
                          uint16 t no planes;
                          uint16_t no_bits_pixel;
                          uint32_t compression;
                          uint32_t size_padding;
                          int32_t x_pixe1_meter;
                          int32 t y pixel meter;
                          uint32 t no colors;
                          uint32 t no imp colors;
                      } image_header;
                      typedef struct {
                          unsigned char R, G, B;
                      } image colors;
                      typedef struct {
                          image_colors *pixels;
                                                        // Imaginea
                      in forma liniarizata
                                                       // Header-ul
                          image header header;
                      imaginii
                          int32 t padding;
                      } image;
                      bool
                             grayscale_image(char*
                                                     path to image,
                      char* path to grey);
                      image load_image(char* path_to_image);
                      参考例程:
                      char *path_to_grey="gray.bmp";
                      grayscale image (inputfile, path to grey);
                      image input, Template;
                      input = load image(path to grey);
                      Template = load image (Templatefile);
void
                      模板匹配, input 是输入的 8 位 BMP 图像, Template
                      是 24 位模板图像, ps 是相似度, ps=0.8,
TemplateMatching2(ch
ar*
                      colors=(image\_colors) \{255, 0, 0\} .
         input, char*
Template, char*
```

```
output, float
ps, image colors
colors)
                      模板匹配, input 是输入的 8 位 BMP 图像, Template
void
TemplateMatching2(im
                      是 24 位模板图像, ps 是相似度, ps=0.8,
                      colors=(image colors) \{255, 0, 0\}.
         input, image
age
                      需引入以下结构体和头文件以及声明:
Template, char*
output, float
                      #include <stdint.h>
                      typedef struct {
ps, image colors
colors)
                          uint16_t signature;
                          uint32 t size;
                          uint16 t reserved1;
                          uint16 t reserved2;
                          uint32_t offset;
                          uint32_t bitmapinfo;
                          int32 t width;
                          int32 t height;
                          uint16 t no planes;
                          uint16_t no_bits_pixe1;
                          uint32 t compression;
                          uint32_t size_padding;
                          int32 t x pixel meter;
                          int32_t y_pixe1_meter;
                          uint32_t no_colors;
                          uint32_t no_imp_colors;
                      } image header;
                      typedef struct {
                          unsigned char R, G, B;
                      } image_colors;
                      typedef struct {
                          image colors *pixels;
                                                       // Imaginea
                      in forma liniarizata
                          image header header;
                                                       // Header-ul
                      imaginii
                          int32_t padding;
                      } image;
                      bool
                             grayscale image(char*
                                                     path to image,
                      char* path to grey);
                      image load image(char* path to image);
                      参考例程:
                      char *path to grey="gray.bmp";
                      grayscale image (inputfile, path to grey);
                      image input, Template;
                      input = load image(path to grey);
```

```
Template = load_image(Templatefile);
                      模板匹配, input 是输入的 8 位 BMP 图像, Template
window
                      是 24 位模板图像, ps 是相似度, ps=0.8,
TemplateMatching4(ch
                     colors = (image colors) \{255, 0, 0\}.
ar*
         input, char*
Template, char*
output, float
ps, image colors
colors)
                      模板匹配, input 是输入的 8 位 BMP 图像, Template
window
                      是 24 位模板图像, ps 是相似度, ps=0.8,
TemplateMatching4(im
                      colors=(image colors) \{255, 0, 0\}.
         input, image
                      需引入以下结构体和头文件以及声明:
Template, char*
output, float
                     #include <stdint.h>
ps, image colors
                      typedef struct {
colors)
                         uint16_t signature;
                         uint32 t size;
                          uint16_t reserved1;
                          uint16 t reserved2;
                         uint32_t offset;
                         uint32 t bitmapinfo;
                          int32 t width;
                          int32_t height;
                         uint16 t no planes;
                         uint16_t no_bits_pixe1;
                         uint32 t compression;
                         uint32_t size_padding;
                          int32 t x pixel meter;
                          int32_t y_pixe1_meter;
                         uint32 t no colors;
                         uint32 t no imp colors;
                      } image header;
                      typedef struct {
                          unsigned char R, G, B;
                      } image colors;
                      typedef struct {
                                                       // Imaginea
                          image colors *pixels;
                      in forma liniarizata
                                                      // Header-ul
                          image header header;
                      imaginii
                          int32_t padding;
                      } image;
                             grayscale image(char*
                     bool
                                                    path to image,
                      char* path_to_grey);
                      image load_image(char* path_to_image);
```

```
参考例程:
                      char *path to grey="gray.bmp";
                      grayscale image (inputfile, path to grey);
                      image input, Template;
                      input = load_image(path_to_grey);
                      Template = load image (Templatefile);
                      模板匹配, input 是输入的 8 位 BMP 图像, Template
window*
                      是 24 位模板图像, ps 是相似度, ps=0.8,
TemplateMatching6(ch
                     colors=(image colors) \{255, 0, 0\}.
ar*
         input, char*
Template, char*
output, float
ps, image_colors
colors)
                      模板匹配, input 是输入的 8 位 BMP 图像, Template
window*
TemplateMatching6(im
                      是 24 位模板图像, ps 是相似度, ps=0.8,
                      colors = (image colors) \{255, 0, 0\}.
age
         input, image
                      需引入以下结构体和头文件以及声明:
Template, char*
output, float
                     #include <stdint.h>
ps, image colors
                      typedef struct {
colors)
                         uint16 t signature;
                         uint32 t size;
                          uint16_t reserved1;
                         uint16_t reserved2;
                         uint32_t offset;
                         uint32 t bitmapinfo;
                          int32_t width;
                          int32 t height;
                         uint16_t no_planes;
                         uint16_t no_bits_pixel;
                         uint32 t compression;
                          uint32_t size_padding;
                          int32 t x pixel meter;
                          int32_t y_pixe1_meter;
                         uint32 t no colors;
                         uint32 t no imp colors;
                      } image header;
                      typedef struct {
                          unsigned char R, G, B;
                      } image colors;
                      typedef struct {
                          image colors *pixels;
                                                       // Imaginea
                      in forma liniarizata
                          image_header header;
                                                      // Header-ul
                      imaginii
```

```
int32_t padding;
                   } image;
                   bool
                         grayscale image(char*
                                               path to image,
                   char* path to grey);
                   image load image (char* path to image);
                   参考例程:
                   char *path to grey="gray.bmp";
                   grayscale image (inputfile, path to grey);
                   image input, Template;
                   input = load image(path to grey);
                   Template = load image (Templatefile);
                   模板匹配,返回值中第一个元素是匹配框左上角的纵
int*
TemplateMatching(cha
                   坐标,第二个元素是匹配框左下角的纵坐标,第三个
                   元素是匹配框的左上角的横坐标,第四个元素是匹配
       input1, char*
                   框的右上角的横坐标。input1 是搜索图像, input2
input2, char*
                   是模板图像, output 是匹配结果图像,
output, unsigned char
red, unsigned
                   MatchScore=0.9.
              char
green, unsigned
              char
blue, double
MatchScore)
                   模板匹配,返回值中第一个元素是匹配框左上角的纵
int*
TemplateMatching(cha
                   坐标,第二个元素是匹配框左下角的纵坐标,第三个
                   元素是匹配框的左上角的横坐标,第四个元素是匹配
       input1, char*
                   框的右上角的横坐标。input1 是搜索图像, input2
input2, unsigned char
                   是模板图像, output 是匹配结果图像,
red, unsigned
              char
green, unsigned
                   MatchScore=0.9.
              char
blue, double
MatchScore)
                   模板匹配,返回匹配结果的图像,input1 是搜索图
struct
            imagine
                   像, input2 是模板图像, MatchScore=0.9。需引入以
TemplateMatching(str
                    下结构体:
            imagine
                   typedef struct imagine {
ColorSource, struct
imagine
                       unsigned char *R, *G, *B, *header;
input1, struct
                       int W, H, Wpad, size;
                   };
imagine
input2, unsigned char
                   声明:
red, unsigned
              char
                                         grayscale image(char*
                   void
green, unsigned
              char
                   nume_fisier_sursa, char*
blue, double
                   nume fisier destinatie);
MatchScore)
                                                       (char
                   struct
                             imagine
                                       salvareBitmap
                   *destinatieFisier):
                   void afisare
                                  (struct
                                          imagine
                                                  img,
                                                        char
                   *destinatieSalvare);
                   将灰度应用于所有图像:
```

```
struct imagine output;
                        grayscale image(inputImage1,
                     input1 grayscale);
                         grayscale image(inputImage2,
                     input2_grayscale);
                        ColorSource=salvareBitmap(inputImage1);
                     //inputImagel 是彩色原图
                         input1=salvareBitmap(input1 grayscale);
                         input2=salvareBitmap(input2 grayscale);
                     output=TemplateMatching(ColorSource, input1, in
                     put2, red, green, blue, MatchScore);
                        afisare (output, outputfile);
                     模板匹配,返回值中第一个元素是匹配框左上角的纵
int*
                     坐标,第二个元素是匹配框左下角的纵坐标,第三个
TemplateMatching(str
                     元素是匹配框的左上角的横坐标,第四个元素是匹配
uct
            imagine
                     框的右上角的横坐标。input1 是搜索图像, input2
input1, struct
                     是模板图像, output 是匹配结果图像,
imagine
input2, unsigned char
                     MatchScore=0.9.
red, unsigned
                     需引入以下结构体:
               char
                     typedef struct imagine {
green, unsigned
               char
blue, double
                        unsigned char *R, *G, *B, *header;
MatchScore)
                         int W, H, Wpad, size;
                     };
                     声明:
                     void
                                           grayscale image(char*
                     nume fisier sursa, char*
                     nume fisier destinatie);
                     struct
                                          salvareBitmap
                                                           (char
                               imagine
                     *destinatieFisier):
                     将灰度应用于所有图像:
                        struct imagine input1, input2;
                        grayscale image(inputImage1,
                     inputl_grayscale);
                         grayscale_image(inputImage2,
                     input2_grayscale);
                        input1=salvareBitmap(input1_grayscale);
                        input2=salvareBitmap(input2 grayscale);
void
                     模板匹配, suprapunereMaxima 表示最大重叠率,参
TemplateMatching(cha
                     考:MaximumMatchingQuantity=10,MatchScore=0.8,
        input, char*
                     suprapunereMaxima=0.2.
templatename, char*
output, unsigned
MaximumMatchingQuant
```

ity, double	
MatchScore, float	
suprapunereMaxima, un	
signed char	
red, unsigned char	
green, unsigned char	
blue)	
int*	模板匹配,支持24位BMP图像,返回匹配框的左上
TemplateMatching(cha	角坐标 (x, y)。
r* input, char*	
Template)	
int*	模板匹配,支持 24 位 BMP 图像,返回匹配框的左上
TemplateMatching(bmp	角坐标 (x, y)。
_img input,bmp_img	需引入以下结构体和声明:
Template)	enum bmp_error
	{
	BMP_FILE_NOT_OPENED = -4,
	BMP_HEADER_NOT_INITIALIZED,
	BMP_INVALID_FILE,
	BMP_ERROR,
	$BMP_OK = 0$
	};
	<pre>typedef struct _bmp_img {</pre>
	bmp_header img_header;
	<pre>bmp_pixel **img_pixels;</pre>
	<pre>} bmp_img;</pre>
	声明:
	enum bmp_error bmp_img_read(bmp_img *, const
	char *);
	参考例程:
	<pre>bmp_img input, Template;</pre>
	<pre>bmp_img_read(&input, inputfile);</pre>
	bmp_img_read(&Template, outputfile);
int*	模板匹配,返回值是匹配框的左上角坐标(x,y)。
TemplateMatching(cha	
r* input1, char*	
input2, char* output)	
int*	模板匹配,返回值是匹配框的左上角坐标(x,y)。
TemplateMatching1(ch	
ar* input1, char*	
input2)	
int* TownlatoMatching?(ch	模板匹配,返回值是匹配框的左上角坐标(x,y)。
TemplateMatching2(ch	

```
input1, char*
ar*
input2)
my_image_comp*
                      模板匹配,返回匹配结果。
TemplateMatching (my
                      需引入以下结构体:
image_comp
                      struct my_image_comp {
input, my_image_comp
                          int width;
Template, int
                          int height;
               H, int
length, float* hpf)
                          int stride;
                          int border:
                          float *handle;
                          float *buf;
                          my_image_comp()
                            { width = height = stride = border = 0;
                      handle = buf = NULL; }
                          my image comp()
                            { if (handle != NULL) delete[] handle; }
                          void init(int height, int width,
                      border)
                              this->width = width; this->height =
                      height;
                              this->border = border;
                              stride = width + 2*border;
                              if (handle != NULL)
                                delete[] handle;
                              handle
                                                                new
                      float[stride*(height+2*border)];
                              buf = handle + (border*stride) +
                      border:
                          void perform_boundary_extension();
                      };
                      struct filt {
                          float* centre;
                          int length;
                      };
                      声明:
                      int read bmp(char*
                                            image,
                                                     my image comp*
                      input comps, int* num comps, int H);
                           write bmp (my image comp* output comps,
                      int
                      char* dest);
                      filt make filter(int type);
                      参考:
                      my_image_comp input;
                      my_image_comp Template;
```

```
filt filter = make_filter(1);
                      int length = filter.length;
                      float* hpf = filter.centre;
                      int H = (filter. length - 1) / 2;
                      int num_comps = 1;
                      read bmp(inputfile, &input, &num comps, 0);
                      read bmp (Templatefile, &Template, &num comps,
                      H);
                      write bmp(&input, outputfile);
                      模板匹配,返回值是匹配框的左上角坐标(x,y)。
int*
TemplateMatching1 (my
                      需引入以下结构体:
                      struct my image comp {
_image_comp
input, my image comp
                          int width;
Template, int
               H, int
                          int height;
length, float* hpf)
                          int stride;
                          int border;
                          float *handle;
                          float *buf;
                          my_image_comp()
                            { width = height = stride = border = 0;
                      handle = buf = NULL; }
                          ~my image comp()
                            { if (handle != NULL) delete[] handle; }
                          void init(int height, int width,
                      border)
                              this->width = width; this->height =
                      height: this->border = border;
                              stride = width + 2*border;
                              if (handle != NULL)
                                delete[] handle;
                              handle
                                                                new
                      float[stride*(height+2*border)];
                              buf = handle + (border*stride)
                      border:
                          void perform boundary extension();
                      };
                      struct filt {
                          float* centre;
                          int length;
                      };
                      声明:
                      int
                           read bmp(char*
                                            image,
                                                     my image comp*
```

```
input_comps, int* num_comps, int H);
                           write bmp (my image comp* output comps,
                      char* dest);
                      filt make filter(int type);
                      参考:
                      my image comp input;
                      my image comp Template;
                      filt filter = make_filter(1);
                      int length = filter.length;
                      float* hpf = filter.centre;
                      int H = (filter. length - 1) / 2;
                      int num comps = 1;
                      read bmp(inputfile, &input, &num comps, 0);
                      read bmp (Templatefile, &Template, &num comps,
                      H);
                      write_bmp(&input, outputfile);
                      模板匹配,返回值是匹配框的左上角坐标(x,y)。
int*
TemplateMatching2(my
                      需引入以下结构体:
_image_comp
                      struct my_image_comp {
input, my image comp
                          int width;
Template, int
               H, int
                          int height:
length, float* hpf)
                          int stride;
                          int border:
                          float *handle;
                          float *buf;
                          my_image_comp()
                            { width = height = stride = border = 0;
                      handle = buf = NULL; }
                          ~my_image_comp()
                            { if (handle != NULL) delete[] handle; }
                          void init(int height, int width,
                      border)
                              this->width = width;
                                                     this->height =
                      height:
                              this->border = border:
                              stride = width + 2*border;
                              if (handle != NULL)
                                delete[] handle;
                              handle
                                                                new
                      float[stride*(height+2*border)];
                              buf = handle + (border*stride) +
                      border:
                          void perform boundary extension();
```

```
};
                     struct filt {
                         float* centre;
                         int length;
                     };
                     声明:
                     int
                          read bmp(char*
                                           image,
                                                   my image comp*
                     input comps, int* num comps, int H);
                          write bmp (my image comp*
                                                    output comps,
                     int
                     char* dest);
                     filt make filter(int type);
                     参考:
                     my image comp input;
                     my image comp Template;
                     filt filter = make_filter(1);
                     int length = filter.length;
                     float* hpf = filter.centre;
                     int H = (filter. length - 1) / 2;
                     int num comps = 1;
                     read_bmp(inputfile, &input, &num_comps, 0);
                     read bmp (Templatefile, &Template, &num comps,
                     H);
                     write bmp(&input, outputfile);
                     模板匹配,返回值是匹配框的左上角坐标(x,y)。
int*
TemplateMatching(cha
                     min 是与匹配分数相关的参数,参考: min=65026。
        input1, char*
input2, char*
output, float min)
                     模板匹配,返回值是匹配框的左上角坐标(x,y)。
int*
TemplateMatching(bmp
                     min 是与匹配分数相关的参数,参考: min=65026。
                     需引入以下结构体:
       input1, bmp in
input2, float min)
                     struct bmp in {
                         int num_components, rows, cols;
                         int num unread rows;
                         int line_bytes;
                         int alignment bytes;
                         FILE *in;
                     };
                     声明:
                     extern int bmp_in_open(bmp_in *state, const
                     char *fname);
                     参考:
                     bmp_in input1, input2;
                     bmp_in_open(&input1, input1file);
```

bmp_in_open(&input2, input2file); 模板匹配,支持 JPG 图像,返回值的第1和第2个元 double* 素是匹配框左上角顶点的横坐标和纵坐标,第3个元 TemplateMatch(byte** input, byte*** 素是目标与模板相对的旋转角度,第4个元素是缩放 Template, char* 比例。参考: size=1, best loss=1000000000, a=0.5, b=2.1, c=0.5, d=45, e1=20, e2=20. output, int irows, int 声明: icols, int trows, int tcols, int size, int #define byte unsigned char best loss, double byte ***LoadRgb (const char *fname, int *rows, int *cols, int *chan); a, double b, double c, double d. int 参考: e1, int e2) int irows, icols, ichan; int trows, tcols, tchan; byte*** input = LoadRgb (inputfile, &irows, &icols, &ichan); byte*** Template = LoadRgb(templatefile, &trows, &tcols, &tchan); double* 模板匹配,支持 JPG 图像,返回值的第1和第2个元 素是匹配框左上角顶点的横坐标和纵坐标,第3个元 TemplateMatch(byte** 素是目标与模板相对的旋转角度,第4个元素是缩放 input, byte*** Template, int 比例。参考: size=1, best loss=1000000000, a=0.5, irows, int icols, int b=2.1, c=0.5, d=45, e1=20, e2=20. 声明: trows, int tcols, int size, int #define byte unsigned char best loss, double byte ***LoadRgb(const char *fname, int *rows, a, double b, double int *cols, int *chan); d, int 参考: c, double e1, int e2) int irows, icols, ichan; int trows, tcols, tchan; byte*** input = LoadRgb (inputfile, &irows, &icols, &ichan); byte*** Template = LoadRgb(templatefile, &trows, &tcols, &tchan); byte*** 模板匹配, 支持 JPG 图像, 返回匹配结果。参考: size=1, best loss=1000000000, a=0.5, b=2.1, TemplateMatch1 (byte* ** input, byte*** c=0.5, d=45, e1=20, e2=20. 声明: Template, int irows, int icols, int #define byte unsigned char trows, int tcols, int byte ***LoadRgb (const char *fname, int *rows, size, int int *cols, int *chan); best loss, double void SaveRgbPng(byte ***in, const char *fname, a, double b, double int rows, int cols); c, double d, int 参考: e1, int e2) int irows, icols, ichan;

<pre>int ObjectFind(bmpread_t input, bmpread_t Template)</pre>	int trows, tcols, tchan; byte*** input = LoadRgb(inputfile, &irows, &icols, &ichan); byte*** Template = LoadRgb(templatefile, &trows, &tcols, &tchan); 模板匹配, 返回匹配到的目标数量。 需引入以下头文件: #include "bmpread1.h" 参考: bmpread_t input, Template; bmpread(inputfile, BMPREAD_BYTE_ALIGN BMPREAD_ANY_SIZE, &input); bmpread(Templatefile, BMPREAD_BYTE_ALIGN BMPREAD_ANY_SIZE, &Template);
float* ImageMatching(char* TargetImage, char* Template0, char* Template1, char* Template2, char* Template3, char* Template4, char* Template5, char* Template6, char* Template7, char* Template8, char* Template8, char* Template9)	图像匹配,返回值中的前 10 个元素是按顺序对应的目标与模板的差异分数,最后一个元素是匹配到的模板的序号。
float* ImageMatching(char* TargetImage, char* Template0, char* Template1)	图像匹配,返回值中的前2个元素是按顺序对应的目标与模板的差异分数,最后一个元素是匹配到的模板的序号。支持BMP图像。
float* ImageMatching(Image3 TargetImage, float **templates, int num_templates)	图像匹配,如果模板数为 n,则返回值中的前 n 个元素是按顺序对应的目标与模板的差异分数,最后一个元素是匹配到的模板的序号。需引入以下结构体: typedef struct { float *data; int width; int height; } Image3; 声明:

```
Image3 load bmp(char *filename);
                     Image3 img, TargetImage;
                     float **templates;
                     int num_templates = 10;
                        templates = (float **) malloc(sizeof(float
                    *) * num templates);
                        img = load bmp(Template0);
                        templates[0] = img. data;
                        img = load bmp(Template1);
                        templates[1] = img. data;
                        img = load bmp(Template2);
                        templates[2] = img. data;
                        img = load bmp(Template3);
                        templates[3] = img. data;
                        img = load bmp(Template4);
                        templates[4] = img. data;
                        img = load bmp(Template5);
                        templates[5] = img. data;
                        img = load bmp(Template6);
                        templates[6] = img. data;
                        img = load_bmp(Template7);
                        templates[7] = img. data;
                        img = load bmp(Template8);
                        templates[8] = img. data;
                        img = load bmp(Template9);
                        templates[9] = img.data;
                     TargetImage = load bmp(TargetImagefile);
void
                     图像特征。
ImageFeatures(char*
                    kernel 文件内容样例:
input, char*
kernel, char* output)
                     1
                    0 - 1 0
                     -1 5 -1
                    0 - 1 0
                     其中,3表示尺寸为3*3,1表示内核的规模
void FileWrite(char*
                     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
BMP, char* TXT)
                     写入图像的文本文件名。
                     图像隐写之文件写出,将文本文件从图像中取出来。
void
FileWriteOut(char*
                     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。
BMP, char* TXT)
void
          LBP (char*
                    LBP 图像特征提取。支持 PNG 图像。
input, char* output)
```

void Watershed2(char* input, char* inputMarqueurs, char* output, int r, unsigned char R, unsigned char G, unsigned char B) void EcrireImage1(char* input, char*	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 PNG 图像。
output, uint32_t rayon)	
void EcrireImage2(char* input, char* inputMarqueurs, char* output, uint32_t rayon)	图像分割。rayon=5。支持 PNG 图像。
void EcrireLPECouleur1(ch ar* input, char* inputMarqueurs, char* output, uint32_t rayon)	图像分割。rayon=5。支持 PNG 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 PNG 图像。
void EcrireImage3(char* input, char* inputMarqueurs, char* output, uint16_t rayon)	图像分割。rayon=1。支持 PNG 图像。
void EcrireImageCouleursA leatoires(char* input, char* inputMarqueurs, char* output, uint8_t r, uint8_t g, uint8_t	图像分割。rayon=1。支持 PNG 图像。

b, uint16_t rayon)	
void Watershed(char* input, char* inputMarqueurs, char* output, uint8_t r, uint8_t g, uint8_t b, uint8_t a, uint16_t rayon) void FloodFill(char*	图像分割。inputMarqueurs 是输入图像的标记图像。 a 一般为 255, rayon=1。支持 PNG 图像。 图像分割之漫水填充法。x=0, y=0, novaCor=127。
<pre>input, char* output, int</pre>	支持 PGM 文件。
void ConvertCoordinatesTo Graphics(char* input, char* output, double IMAGE_SIZE, double PIXEL_PADDING, bool drawlines)	读取点坐标并输出点之间绘制的点或线段的图像。 IMAGE_SIZE=800, PIXEL_PADDING=25, drawlines=0 或 drawlines=1。 输入文件格式 如果我们将"N"表示为点数,则假设采用以下点坐标文件格式: N 1 x 坐标 y 坐标 2 x 坐标 y 坐标 3 x 坐标 y 坐标 4 x 坐标 y 坐标 N x 坐标 y 坐标
void HumanDetection1(char * input, char* output, double MINH, double MAXH, double MINS, double MAXS)	人类检测。参考: MINH=0.0, MAXH=50.0, MINS=0.23, MAXS=0.68。支持 24 位 BMP 图像。
void HumanDetection2(char * input, char* output, double MINH, double MAXH, double MINS, double MAXS)	人类检测。参考: MINH=0.0, MAXH=50.0, MINS=0.23, MAXS=0.68。 支持 24 位 BMP 图像。
<pre>void HumanDetection3(char * input, char* output, double MINH, double MAXH, double MINS, double MAXS)</pre>	人类检测。参考: MINH=0.0, MAXH=50.0, MINS=0.23, MAXS=0.68。 支持 24 位 BMP 图像。

int ImageFeatureNumber(c har* input)	计算图像的特征点数量。支持 24 位 B	MP 图像。	
<pre>int ContentSimilarity(ch ar* input1, char* input2, int KDTREE_BBF_MAX_NN_CH KS, doub1e NN_SQ_DIST_RATIO_THR)</pre>	返回两个图像的内容相似度。 KDTREE_BBF_MAX_NN_CHKS 是在BBF 搜索期间要检查的关键点 NN 候选的最大数量,NN_SQ_DIST_RATIO_THR 是 NN 和第二个 NN 之间距离平方比的阈值,参考: KDTREE_BBF_MAX_NN_CHKS=100, NN_SQ_DIST_RATIO_THR=0.49。支持24位BMP图像。		
Feature* ImageFeature(char* input)	返回图像的特征数据。支持 24 位 BMP 图像。 需引入一下结构体: typedef struct Point2D64f { double x; double y; }Point2D64f;		
	typedef struct feature { double x; double y; double a;	/x 坐标/ /y 坐标/ /0xford 型	
	仿射区域参数/ double b; 仿射区域参数/	/0xford 型	
	double c; 仿射区域参数/ double scl;	/0xford 型 /Lowe 风格	
	特征的比例/ double ori; 特征的方向/	/Lowe 风格	
	int d; 度/ double descr[128];	/描述符长	
	int type; 型, OXFD或 LOWE/	/功能类	
	int category; 类别/ struct feature* fwd_match;	/通用功能 	
	的匹配特征/ struct feature* bck_match; backmward 图像中匹配特征/	/ 从	

	struct feature* mdl_match;	/匹配模型
	中的特征/	
	Point2D64f img_pt;	/图像中的位
	置/	/模型中的位
	Point2D64f mdl_pt; 置/	/ 医至中时位
	void* feature data;	/用户可定
	义数据/	,,,,,,
	}Feature;	
double	字符匹配,支持 BMP 图像,返回值是目标图像匹配到	
CharacterRecognition	的模板文件的序号,如返回值是2则说明图像与序号	
(char*	为 2 (序号从零开始)的模板匹配。	
TargetImage, char*	参考: TemplateFileGroup[]={ "0.	txt", "1. txt",
<pre>TemplateFileGroup[])</pre>	"2. txt", "3. txt", "4. txt", "5. t	txt", "6. txt",
	"7. txt", "8. txt", "9. txt" };	
double	字符匹配,支持 BMP 图像,返回值是	目标图像匹配到
CharacterRecognition	的模板文件的序号,如返回值是2则说明图像与序号	
1(char*	为 2 (序号从零开始) 的模板匹配。	
TargetImage, char*	参考: TemplateFileGroup[]={ "0.	txt", "1. txt",
<pre>TemplateFileGroup[])</pre>	"2. txt", "3. txt", "4. txt", "5. t	txt", "6. txt",
	"7. txt", "8. txt", "9. txt" };	