#### 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
	} 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 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 A1: (1	
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
1	
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
output)	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	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
	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,
threshold, int nbLevels, float a)	a=10。返回 ISNR。
float Repair2(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,
threshold, int nbLevels, float a)	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	nbLevels=3, pas=1, size_filter 是低
size_filter, float var, int	通过滤器的大小, var 是噪声方差,
nb_iterations, int nbLevels, int	nb_iterations 是 Landweber 的迭代
pas, float a, int b)	数。返回 ISNR。

void Repair1(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。
<pre>void Repair2(char* input, char*</pre>	图像修复,a=0.0,M是分解的层次数,
output, int M, float a)	如 M=3。
void MakeNoise1(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

void YUVsuperposition(char* input1, char*	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	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*	yuv444p 直接叠加到
input2, char* output, int width, int	yuv420p 上, 不做转换,
height, unsigned char Y_BLACK, unsigned char	Y_BLACK 、 U_BLACK 和
U_BLACK, unsigned char V_BLACK)	V_BLACK 用于将原图中的黑
	色变成透明,参考:
	Y_BLACK=16, U_BLACK=128,
	V_BLACK=128。
void YUV444toYUV420(char* input, char*	YUV444 转 YUV420, height
output, int height, int width)	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,V_BLACK=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, YUVwidth 和 YUVheight 是输入的 YUV 图像的宽和高,参考: depth=12, mt=true。
void YUVEdgeProcessingHorizontalDirection(char* input, char* output, int width, int height, double k)	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。
void YUVScale(char* input, char* output, int	缩放 yuv420 图像,参考:
inputWidth, int inputHeight, int	inputWidth=1280 ,
outputWidth, int outputHeight)	inputHeight=720 ,
	outputWidth=128 ,
	outputHeight=72。
<pre>void NoiseTreatment(char* input, char*</pre>	YUV 噪声处理。
output, int width, int height, int	
TWICEwidth, int TWICEheight)	
<pre>void NoiseTreatment(char* input, char*</pre>	YUV 噪声处理。
output, int width, int height, int frame, int	
max_frame)	

### RAW 图像处理

unsigned char** RAWRead(char*	读取 RAW 图像。
input, int height, int width)	
void RAWWrite(unsigned char**	保存 RAW 图像。
input, char* output, int height, int	
width)	
void MBVQ(char* input, char*	MBVQ 效果, input 是输入文件名,
output, int width, int height)	output 是输出文件名。width 和
	height 是输出图像的宽和高。
void RAWtoPPM_red(char*	RAW 转为 PPM 后提取红色通道,参考:
input, char* output, int width, int	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
DAW DDW	
void RAWtoPPM_green1(char*	RAW 转为 PPM 后提取绿色 1 通道,参
input, char* output, int width, int	考: width=4096, height=3072,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。
	黑引入以下枚举:
	南ガバタド牧争:   enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
void RAWtoPPM_green2(char*	RAW 转为 PPM 后提取绿色 2 通道,参
input, char* output, int width, int	考: width=4096, height=3072,
height, Debayer Algorithm algo)	algo=NEARESTNEIGHBOUR 或 LINEAR。
	支持 RAW12 格式。

	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
void RAWtoPPM blue(char*	RAW 转为 PPM 后提取蓝色通道,参考:
input, char* output, int width, int	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
1 DAW, DDM/ 1	);
void RAWtoPPM(char* input, char*	RAW 转为 PPM,参考: width=4096,
output, int width, int	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
	};
<pre>void RawPowerTransformation(char*</pre>	幂次变换, input 是输入的 RAW 图像
input, char* output, int width, int	文件名, output 是输出的 RAW 图像文
height, int c, float v)	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*	平均滤波器, input 是输入文件名,
input, char* output, int ROWS, int	output 是输出文件名。ROWS 是图像的
COLS, int M, float mask[3][3])	行大小,COLS 是图像的列大小,M 是
oods, into my riod o mask [o] [o]/	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:
	多名英似:   float
	{{0.1111, 0.1111, 0.1111},
	( (O. 1111, O. 1111, O. 1111),
	\[ \langle 0 \ 1111 \ 0 \ 1111 \\ \ \ \ \ \ \ \ \
	{0. 1111, 0. 1111, 0. 1111},
	[0 1111 0 1111 0 1111]]
.1 D + + / 1	{0.1111, 0.1111, 0.1111}};
void RawImageInversion(char*	图像反相, input 是输入的 RAW 图像
input, char* output, int width, int	文件名, output 是输出的 RAW 图像文
height)	件名, width 是输入图像的宽, height
101010/	是输入图像的高。支持 RAW 图像。

void	直方图均衡化,input 是输入的 RAW
RawHistogramEqualization(char*	图像文件名, output 是输出的 RAW 图
input, char* output, int width, int	像文件名, width 是输入图像的宽,
height)	height 是输入图像的高。支持 RAW 图
	像。
void	RAW 直方图均衡化, width 和 height
RAWHistogramEqualization(char*	是输入图像的宽和高。
input, char* output, int width, int	
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 是图像的
COLS, int M, int sequence[9])	行,COLS 是图像的列,M 是滤波相关
cobs, int m, int sequence[3])	参数,如 M=1。支持 RAW 图像。
	多
	多写侯似:   int
world Downto Down 1 (shows to be seen to be	sequence[9]={0,0,0,0,0,0,0,0,0};
void RawtoBmp1(char* input, char*	RAW 图像转为 BMP 图像, input 是输
output, unsigned long Width,	入文件名,output 是输出文件名。
unsigned long Height)	Width 和 Height 是输入文件的宽和
	高。 
void RawToBmp(char* input, char*	RAW 图像转为 BMP 图像, input 是输
output, int imageWidth, int	入文件名,output 是输出文件名。支
imageHigth)	持宽和高相等的图像。
void RGBtoHSI(char* input, char*	RGB 模型转为 HIS 模型,input 是输
output)	入文件名,output 是输出文件名。支
	持 24 位 BMP 图像。
void CyanGray(char* input, char*	青色灰度图像。
output, int width, int height)	口层长序围绕
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)	11.14 - 714.
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)	th E/ 56 H
void MovieEffect (char* input, char*	电影效果。
output, int width, int height)	
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 TernaryGrayLeve12(char*	三值灰度。
input, char* output, int width, int	
height)	
void BestEdgeMap1(char*	最佳边贴图。
input, char* output, int width, int	
height)	
void BestEdgeMap2(char*	最佳边贴图。
input, char* output, int width, int	
height)	
void Skeletonize(char* input, char*	骨架化。
output, int width, int height)	
void SeparableDiffusion(char*	可分离扩散。
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*	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	
height, int newwidth, int newheight)	
void DitherMatrixMethod(char*	四级抖动,默认 N=2。
input, char* output, int width, int	
The state of the s	

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
height, int N)	
void Dewarped1(char* input, char*	脱蜡。a是在输出图像中检查半径是
output, int width, int height, int	否<=a, 然后再进行扭曲, 参考:
Offset, double a, double b)	Offset=256, a=256.5, b=0.5。
void Dewarped2(char* input, char*	脱蜡。a 是在输出图像中检查半径是
output, int width, int height, int	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 };
	,
	double coeffy[12] = {-
	5. 67763072e-04, 1. 00056888e+00,
	1. 13998357e-03,
	·
	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);
void TextureSegmentation1(char*	纹理分割,默认 K=6, N=100。
input, char* output, int width, int	
height, int K, int N)	Over A deal HIVI as a series
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量,
<pre><string> filename, char* output, int</string></pre>	如 filename 里有 3 个图像名称,则
width, int height, int K, int N, int	a=3; output 是分类结果文件,格式
a)	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。

<pre>input, char* output, int width, int height)</pre>	
void ErrorDiffusion3(char* input, char* output, int width, int	误差扩散。
height) void Thin(char* input, char* output, int width, int height)	图像细化。
<pre>void OilPainting(char* input, char* output, int width, int height, int N)</pre>	油画效果,默认 N=2。
void OilPainting1 (char* input, char* output, int width, int height, int N)	油画效果,默认 N=2。
void AverageFiltering(char* inputfile, char* outputfile, int	3*3 平均值滤波。
<pre>width, int height) void GeometricMeanFiltering(char* inputfile, char* outputfile, int</pre>	3*3 几何均值滤波。
width, int height)  void MedianFiltering (char* inputfile, char* outputfile, int	中值滤波。
<pre>width, int height) void FFT(char* input, char* output, int width, int height)</pre>	FFT 函数。
<pre>void LowPassOrHighPassFiltering(char* input, char* output, int width, int height, int LOW_PASS, int DEGREE)</pre>	低通或高通滤波。LOW_PASS=1 为低通滤波,否则为高通滤波,DEGREE 为滤波程度,如 DEGREE=0。
<pre>void IFFT(char* input, char* output, int width, int height, int LOW_PASS, int DEGREE)</pre>	IFFT 函数。LOW_PASS=1 为低通滤波, 否则为高通滤波,DEGREE 为滤波程 度,如 DEGREE=0。
void BMPtoRAW(char* inputfile, char* outputfile)	BMP 图像转 RAW 图像。支持 24 为 BMP 图像。
<pre>void BMPtoRAW1(char* input, char* output)</pre>	BMP 图像转 RAW 图像。支持 24 为 BMP 图像。

# BMP 图像处理

unsigned char**	读取 8 位 BMP 图像的像素。
BMPRead8(char* input)	
	生成 8 位 BMP 图像, output 是生成的图像
	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
	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;
unsigned int BMPHeight(char*	读取 BMP 图像的高度。
input)	
unsigned int BMPWidth(char*	读取 BMP 图像的宽度。
input)	
void GenerateImage(char*	生成 24 位和 32 位 BMP 图像。type 等于图
output, BMPMat**	像的位数,如 type=24。
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++) {
	for (unsigned int j = 0; j <
	2450; j++) {
	color[i][j].B =0;
	color[i][j].G =0;
	color[i][j].R =255; }
	}
void	直方图均衡,支持 8 位和 16 位 BMP。input
HistogramEqualization5(char*	是输入文件名,output 是输出文件名。
input, char* output)	
void Resize(char*	图片缩放,支持8位和16位BMP。input是
input, char* output, int	输入文件名, output 是输出文件名。Height
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。 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。
00 0.1.1.0.1.(1 1 1 D . 1 P)	4/ 1.1.4 F-1/ 1 H 4 2/G/🔨 °

int matter that (DITE steen)	<b>共俎圆山的</b> 宣庭
int getHeight (FILE *fp)	获得图片的高度。
<pre>unsigned short getBit(FILE *fp)</pre>	获得每个像素的位数。
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,
void PNGIllinify(char*	d=80, e=0.5。 幻化效果, input 是输入文件名, output 是
input, char* output)	新出文件名。
void PNGWaterMark(char*	图像加水印, input1 和 input2 的尺寸必须
input1, char* input2, char*	相同。
output)	AHLA O
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)	actor are Villar E Dur El Mo
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)	
dop on/	

倒立特效。a=1,b=128,c=0.5,depth=24。
支持 24 位 BMP 图像。
744 11 12 Day 12 1340
肥胖特效。a=1, b=128, c=0.5, depth=24。
支持 24 位 BMP 图像。
大孙 21 图 Billi 国家。
高脚特效。a=1, b=128, c=2, d=0.5,
depth=24。支持 24 位 BMP 图像。
depen 210 XII 21 E Dim Ediko
弧度弯曲特效。a=1,b=128,c=4,d=2,e=0.5,
depth=24。支持 24 位 BMP 图像。
细化特效。a=1, b=128, c=0.5, d=0.5,
depth=24。支持 24 位 BMP 图像。
弯曲特效。lim=20, a=1, b=128, c=4, d=5,
e=0.5, depth=24。支持24位BMP图像。
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 图像的红色通道公县
图像的红色通道分量 ungigned shar A: //权限 22 位 PMD 图
unsigned char A; //仅限 32 位 BMP 图 像的 Alpha 通道
ISBN Alpha 通足 }BMPMat;
J DIMI MAC,
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);

	.1 2
	void Conversion24(BMPMat**
	input,BMPMatdouble** output);
	void Conversion24(BMPMatdouble**
	input,BMPMat** output);
<pre>void CrossDenoising(BMPMat**</pre>	十字法剔除孤立像素点。
input, BMPMat** output, double	需引入以下结构体和声明:
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);
	_ · · · · · · · · · · · · · · · · · · ·
	`
• 1	input, BMPMat** output);
void	交叉法剔除孤立像素点。
CrossConnectionDenoising(uns	需引入以下结构体和声明:
igned char** input, unsigned	typedef struct {
char** 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;
	Jum Mat,

```
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**
                              input, BMPMat** output);
void
                              交叉法剔除孤立像素点。
                              需引入以下结构体和声明:
CrossConnectionDenoising (BMP
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;
                              typedef struct {
                                  double B;
                                  double G:
                                  double R:
                                  double A;
                              }BMPMatdouble;
                                     Conversion8 (unsigned
                              void
                                                             char**
                              input, double** output);
                                               Conversion8(double**
                              void
                              input, unsigned char** output);
                              void
                                              Conversion24 (BMPMat**
                              input, BMPMatdouble** output);
                                        Conversion24 (BMPMatdouble**
                              void
                              input, BMPMat** output);
```

	burt 14 Bl BA 201 A 15 E
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;
	1.6.4
	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**
	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;
	<pre>typedef struct {</pre>
	double B;
	·
	double G;

	1 11 D
	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**
	input, BMPMat** output);
void ImageFusion3(char*	多聚焦图像的融合,支持 8 位 BMP 图像。
input1, char* input2, char*	block_height=8 , block_width=8 ,
output, int block_height, int	threshold=1.75。
block_width, double	
threshold)	
void ImageFusion4(char*	多聚焦图像的融合,支持 8 位 BMP 图像。
input1, char* input2, char*	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 Screenshot3(HWND hWnd,	截屏函数。hWnd 是要截屏的窗口句柄,如:
LPCWSTR OutputImage)	GetDesktopWindow();OutputImage 截图名
	称。
void Screenshot1(HWND hWnd,	截屏函数。hWnd 是要截屏的窗口句柄,如:
LPCWSTR OutputImage)	GetDesktopWindow();OutputImage 截图名
Transition of the state of the	称。
void Screenshot2(HWND	截屏函数。hWnd 是要截屏的窗口句柄,如:
hWnd, LPCWSTR OutputImage)	GetDesktopWindow():OutputImage 截图名
Inna, Di onotik outputtinage)	称。
void Dark(char* input, char*	170°   暗调滤镜,参考: ratio=100°
output, int ratio)	H日 yii yiu bu , 多つ; I a (IO-IOO。
void WaveFilter(char*	波浪形变特效滤镜, degree 是滤镜程度(波
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,
voia i inclirittet (cliat*	加亚ル文的双他说, a=U 时工从 DML 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。
Disabbile of the sale	挤压形变特效滤镜, a=0 时生成 BMP 图像,
void PinchFilter(char*	
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
	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 ColorTransfer(char*	颜色转移。
input1, char* input2, char*	
output)	
void GrayImagel(char*	直方图均衡化。
input, char* output)	
void ChannelHisteq(char*	直方图均衡化。
input, char* output)	
void HSVtoRGB(char* input,	HSV 转到 RGB。
char* output)	
void	直方图均衡化。
HistogramEqualizationOnGrayI	
mage(string input, char*	
output)	
CImg <unsigned int=""></unsigned>	直方图均衡化。
HistogramEqualizationOnGrayI	
mage2(string input)	
void	直方图均衡化。
HistEqualColorImageOneColorC	

hannel(string input, char*	
output)	
CImg <unsigned int=""></unsigned>	直方图均衡化。
HistEqualColorImageOneColorC	
hannell(string input)	
void	直方图均衡化。
HistEqualColorImageThreeColo	
rChannels(string input,	
char* output)	
CImg <unsigned int=""></unsigned>	直方图均衡化。
HistEqualColorImageThreeColo	
rChannels(string input)	
void	HSI 空间。
HistEqualColorImageHSISpace(	
string input, char* output)	
CImg unsigned int>	   HSI 空间。
HistEqualColorImageHSISpace(	101 11.40
string input)	
void ColorTransfer1(char*	颜色转移。
sourceImage, string	
targetImage, char* output)	
CImg <unsigned int=""></unsigned>	
ColorTransfer2(string	
sourceImage, string	
targetImage)	
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
input, onar · output, int a)	程度,数字越大,压缩后的文件体积越小,
	如 a=100。
void	部分颜色保留滤镜,参考: ratio=60。
PartialColorRetention(char*	PLANCE IN THE PROPERTY SO J. LAUTO CO.
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。
515mod ondi . Input, unsigned	

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	一次反图像垂直说像,Input 是拥入图像的像 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	是相八国家的见,II 是相八国家的问。
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	人们,人口以口,几个一个人口以口问。
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 是初始國
barota de la constanta de la c	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。 一 佑 図 梅 府 畑
void Corrosion(char*	二 值 图 像 腐 蚀 , 参 考 :
input, char* output, unsigned	$[\max [9] = \{0, 255, 0, 255, 255, 255, 0, 255, 0\}$

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。
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 <sub>°</sub>
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].
	1

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	7170
*Kernal, int kernalW, int	
halfKernalW)	
	<b>网络</b> 鸭W
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	
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	— <u>ш/м</u> кд v
*output, int rows, int	
cols, int mat[5][5])	
	二值膨胀。
void Expansion (unsigned char	──   □□
*input, unsigned char	
*output, int rows, int	
cols, int mat[5][5])	支票法数 土柱 NVO 之世
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.
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, 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)	<b>七</b> 房 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □
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。
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)	计算思加古方图光咖卧 ;,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
void	计算累加直方图并映射, input 是输入文件
CalculateCumulativeHistogram Map(char* input, char*	名,outfile 是输出文件名。支持 24 位 BMP
i wan ichar* inniit char*	图像。

outfile)	
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** input, unsigned char**	float **转 unsigned char**, output 用于保存结果(与 input 大小相同)。
output) void Conversion8(unsigned char** input, double** output)	unsigned char**转 double **, output 用于保存结果(与 input 大小相同)。
void Conversion8(double** input, unsigned char** output)	double **转 unsigned char**, output 用于保存结果 (与 input 大小相同)。
void Conversion8(unsigned char** input, char** output)	unsigned char**转 char **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion8(char** input, unsigned char** output)</pre>	char **转 unsigned char**,output 用于保存结果(与 input 大小相同)。
<pre>void Conversion24(BMPMat** input, BMPMatshort** output)</pre>	BMPMat **转 BMPMatshort **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion24(BMPMatshort** input, BMPMat** output)</pre>	BMPMatshort **转 BMPMat **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion24(BMPMat** input, BMPMatint** output)</pre>	BMPMat **转 BMPMatint **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion24(BMPMatint** input, BMPMat** output)</pre>	BMPMatint **转 BMPMat **, output 用于保存结果 (与 input 大小相同)。
<pre>void Conversion24(BMPMat** input,BMPMatfloat** output)</pre>	BMPMat **转 BMPMatfloat **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion24(BMPMatfloat** input, BMPMat** output)</pre>	BMPMatfloat **转 BMPMat **, output 用于保存结果(与 input 大小相同)。
void Conversion24(BMPMat** input, BMPMatdouble** output)	BMPMat **转 BMPMatdouble **, output 用于保存结果 (与 input 大小相同)。
<pre>void Conversion24(BMPMatdouble** input, BMPMat** output)</pre>	BMPMatdouble **转 BMPMat **, output 用于保存结果 (与 input 大小相同)。
<pre>void Conversion24(BMPMat** input,BMPMatchar** output)</pre>	BMPMat **转 BMPMatchar **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion24(BMPMatchar** input, BMPMat** output)</pre>	BMPMatchar **转 BMPMat **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion32(BMPMat** input,BMPMatshort** output)</pre>	BMPMat **转 BMPMatshort **, output 用于保存结果(与 input 大小相同)。
<pre>void Conversion32(BMPMatshort** input, BMPMat** output)</pre>	BMPMatshort **转 BMPMat **, output 用于保存结果(与 input 大小相同)。

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)	<u> </u>
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
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	J. 10.00 30. 2 0
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:
cooling [0], i i out cool)	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
vora imager it terring (char-	国 图 link

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] = $\{\{1, 2, 1\},$
	$\{2, 4, 2\},$
	$\{1, 2, 1\}\};$
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* input, char* output, short	输出文件名。支持 8 位 BMP 图像。 参考模板:
average[3][3], short	short average[3][3] = {{1, 2, 1},
soble1[3][3], short	$\{2, 4, 2\},\$
sob1e2[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)	
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)	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)	output Amul All 10 St. O D Bin Elike
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 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
void MosaicFilter(char*	的大小。支持 24 位 BMP 图像。 马赛克滤镜,参考: ratio=50。
input, char* output, int	一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一
ratio)	
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。 宣牝亚妈 : nout 目於) 文件权 out nut 目
void GaussSmoothSharpen(char*	高斯平滑, input 是输入文件名, output 是 输出文件名。Template 是高斯平滑模板,
input, char* output, int	加山文行名。 Template た同朔「稍候放,」 coefficient=16。支持 24 位 BMP 图像。
Template[3][3], int	200111010110 100 文刊 71 压 Dim 国队。
coefficient)	
void SobelSharpen(char*	Sobel 算子, input 是输入文件名, output
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)	
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
input, char* output)	输出文件名。支持 8 位 BMP 图像。
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。
mode1[9])	
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, unsigned char	threshold=115; color1 和 color2 是要填
color1, unsigned char color2)	充的两个颜色。支持 24 位 BMP 图像。
void	filename 是生成的 PNG 图像文件名; img 是
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}, r=50, a=0.5, m=1。支持 RAS 文件。
void MakeSphere(double	使用反射模型生成球体的图像, vector_v
vector_v[3], double	是摄影机的方向, output 是输出的结果图
vector_s[3], double r, double	像文件名,ROWS 是输出图像的行数,COLS
a, double m, int ROWS, int	是输出图像的列数,参考: vector_v[3] =
COLS, char* output, double	$\{0.0, 0.0, 1.0\}, \text{ vector\_s}[3] = \{0.0, 0.0, 0.0\}$
max)	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	具有圆形结构集的双层形态侵蚀,支持8位
DoubleLayerErosion(char*	和 24 位 BMP 图像。
input, char* output)	
void	二值图像水平镜像。
BinaryImageHorizontalMirror(	
unsigned char	
*input, unsigned char	
*output, unsigned int	
w, unsigned int h)	
void	灰阶图像水平镜像。
GrayImageHorizontalMirror(un	
signed char *input, unsigned	
char *output, unsigned int	
w, unsigned int h)	
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	<b>秋山山地</b>
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)	interpolation of interpolation 18
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)	
	   通道分离,input 是输入文件名,Routput
void ChannelSeparation(char*	是红色通道图像,Goutput 是绿色通道图
input, char* Routput, char*	
Goutput, char* Boutput)	像,Boutput 是绿色通道图像。支持 24 位 pmp 图像
world DottomM-th-1/-h	BMP 图像。 图安注 input 見給 ) 立伏夕 output 見給
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> &amp;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: 输入图像
char *src, unsigned char	Lin]
	_
*guidance, unsigned char	[in/out]] dst: 输出图像
*dst, int w, int h, int c, int	[in] w: 宽
r, float sigma_s, float	[in] h: 高
sigma_r)	[in] c: 图像通道,仅 c=1 或 c=3
	[in] r: 局部窗口半径
	[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。
	@return: 0:ok; 1:error
int BoxfilterFilter(unsigned	方盒滤波。
char *src, unsigned char	[in] src: 输入图像,单通道
*dst, int w, int h, int c, int	[in/out] dst: 输出图像,单通道
r)	[in] w: 宽
	[in] h: 高
	[in] c: 图像通道,仅 c=1
	[in] r: 局部窗口半径
	return: 0:ok; 1:error
int	方盒滤波。
BoxfilterFilter1(unsigned	[in] src: 输入图像,单通道
char *src, unsigned char	
*dst, int w, int h, int c, int	[in] w: 宽
r)	lin] h: 高
	[in] c: 图像通道,仅 c=1
	[in] r: 局部窗口半径
	return: 0:ok; 1:error
int	快速导向滤波
fast_guided_filter(unsigned	[in] src: 输入图像,单通道
char *src, unsigned char	[in] guidance: 引导图像,单通道
*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]   w.
scale)	
	[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

	龙台·
	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	
int	快速导向滤波
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
	足住的高國恒   [in] HoughRho: 累加器的距离分辨率(以
_ ••• ,	_
bbox,	像素为单位)
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]
                                  с:
                         [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: 输入图像
_	
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,	[in] h: 高
float sigma_r, float _scale)	[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
	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	同心心权部,多与,preserve o。
preserve)	
void EmbossFilter(char*	浮雕过滤器,参考: preserve=1。
input, char* output, int	11 MEKENBAR 9 9. Preserve 18
preserve)	
void SharpenFilter(char*	锐化过滤器,参考: preserve=1。
input, char* output, int	3. Farana a
preserve)	
void Convolution(char*	卷积,参考: w=7, preserve=1。
input, char* output, int w, int	_ ,
preserve)	
void GaussianBlur(char*	高斯模糊,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	
void HybridImage(char*	混合图像,参考: sigma=2, preserve=1。
input1, char* input2, char*	
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 皮肤过滤器,a=2,b=1。
<pre>input, char* output, int a, int b)</pre>	/文川八尺-(心伯)
void Resize1(char*	图像模糊, w=713, h=467。
input, char* output, int w, int	国
h)	
void Resize2(char*	图像模糊。
input, char* output, int w, int	F-1 KW 174,104 o
impac, onar. output, int w, int	

1.)	
h)	
void Shift(char* input, char*	Shift函数, ch=1, v=0.1。
output, int ch, float v)	DOD ## HOL
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 ColorTransfer(char*	颜色转移。
input1, char* input2, char*	
output)	
void DrawText(char*	文本绘制,R=255,G=255,B=255,depth=1,
<pre>inputText, char* output, int</pre>	spectrum=3, (x,y)是文本的坐标, color1
width, int height, int	是前景颜色, color2 是背景颜色,
depth, int spectrum, int x, int	opacity=1, font=60.
y, unsigned char R,unsigned	
char G, unsigned char	
B, unsigned char	
color1[], unsigned char	
color2[], float	
opacity, unsigned int font)	
void EqualizedGray(char*	灰度图直方图均衡化。
input, char* output)	
void	彩色图直方图均衡化。
ColorHistogramEqualization(c	
har* input, char* output)	
void AverageHistogram(char*	直方图均衡化。
input, char* output)	
void HSIHist(char* input,	HIS 直方图。
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)	
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 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 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 tresholdl, 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	
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。
`	凶脉火亡。
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	7,130,21,70, 19,10,100
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*	点为因 <b>约</b> 岗化。
* input, short** output, long	
height, long width, short	
GRAY LEVELS)	
void	平滑线性滤波器。
SmoothLinearFiltering(short*	
* input, short** output, long	
height, long width, short	
average[3][3])	
void MedianFiltering(short**	中值滤波器。
input, short** output, long	The work and the second
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**	
input, short** output, long	
height, long width)	
void MeanFilter(short**	均值滤波器。
input, short** output, long	
height, long width)	
void	几何均值滤波器,默认 product=1.0。
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。
L	- · · · · · · · · · · · · · · · · · · ·

W	
MatrixGlobalDivision24(BMPMa	
t** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相加。
MatrixGlobalAddition32(BMPMa	
t** input1, BMPMat**	
input2,BMPMat** output)	
void	基于模板矩阵的全局相减。
MatrixGlobalSubtraction32(BM	
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	V(V() E1   RV = / V   R /V
t** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相加。
MatrixGlobalAddition8(unsign	V(V() E1   RV = / V   R/VIII
ed char** input1, unsigned	
char** input2, unsigned	
char** output)	
void	基于模板矩阵的全局相减。
MatrixGlobalSubtraction8(uns	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
igned char** input1, unsigned	
char** input2, unsigned	
char** output)	
void	基于模板矩阵的全局相乘。
MatrixGlobalMultiplication8(	THE TOTAL PROPERTY OF THE PERTY
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	《A2,,927 是是形的石下角的主体。   函数源代码:
Output, Int XI, IIIt yI, IIIt	四双切引门时:

```
需引入以下头文件:
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);
                                                   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 < height; i++) {
                                        for (unsigned
                                                          int
                                                                 j
                                0; j \le (idth; 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);
void
                              灰度图以矩形方式局部截取且其它部分填
                              充,(x1,y1)是矩形的左上角的坐标,
GrayRectangleLocalSegmentati
on (char*
                 input, char*
                               (x2, y2) 是矩形的右下角的坐标。
output, int
             x1, int
                      v1, 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
                                                     BMPRead8(char*
                                           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:
                                  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
                              v1:i \le v2: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
ColorDrawRectangle(char*
                              的坐标,(x2, y2)是矩形的右下角的坐标。
input, char*
                              函数源代码:
                  output, int
                              需引入以下头文件:
x1, int
           y1, 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 < height; i++) {
```

```
for (unsigned
                                                         int
                                0; j \le (idth; j++) {
                                           output[i][j]. B=color. B;
                                           output[i][j]. G=color. G;
                                           output[i][j]. R=color. R;
                                    for (unsigned
                                                               i
                                                      int
                                0; i < height; i++) {
                                        for (unsigned
                                                         int
                                                                 j
                                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:
x1, int
           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:
unsigned
                         BMPRead8(char*
             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:
    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 \le (idth; j++) {
           output[i][j]=color;
    for (unsigned
                       int
                                i
0; i < 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);
                              浮雕效果,默认 value=128。
             Relief(BMPMat**
void
input, BMPMat**
                  output, int
value)
void Relief(unsigned char**
                              浮雕效果,默认 value=128。
input, unsigned
                      char**
output, int value)
                              图像锐化,默认 degree=0.3。
         Sharpening (BMPMat**
input, BMPMat** output, double
degree)
                              图像锐化,默认 degree=0.3。
         Sharpening (unsigned
void
char** input, unsigned char**
output, double degree)
                              图像柔化,默认 value=9。
             Soften (BMPMat**
void
input, BMPMat**
                  output, int
value)
                              图像柔化,默认 value=9。
void Soften(unsigned char**
input, unsigned
                      char**
output, int value)
void flipX(char* input, char*
                              X方向翻转,支持 JPG 文件。
output)
                              Y方向翻转,支持 JPG 文件。
void flipY(char* input, char*
output)
void Crop(char* input, char*
                              裁剪。
output, uint16_t
                    start_x,
uint16_t start_y,
                    uint16 t
new_height,
                    uint16 t
new width)
                Resize(char*
                              缩放。
void
input, char*
                  output, int
new_width, int new_height)
void Scale(char* input, char*
                              比例。
```

output, double ratio)	大庆亚石井
void GrayscaleAvg(char*	灰度平均值。
input, char* output)	<b>卡皮克皮</b>
void grayscaleLum(char*	灰度亮度。
input, char* output)	ぶ/ カ ) ☆ 四
void ColorMask(char*	彩色遮罩。
input, char* output, float	
r, float g, float b)	<u> </u>
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	
strength)	
void EdgeDetection(char*	边缘检测,参考: cutoff=115。
input, char* output, double	
cutoff)	を行わ
void Sharpen(char*	锐化。
input, char* output)	
void CannyProcessing(char*	Canny 处理, a 可以为 1、2、3、4、5。支持
input, char* output, int a)	BMP图像。
void AverageGrayScale(char*	平均灰度化。
input, char* output)	☆ 日 pw
void SimpleBW(char*	简易 BW。
input, char* output)	⇒/刃 pw
void AdvancedBW(char*	高级 BW。
input, char* output)	44.51喝 主
void UniformNoise(char*	均匀噪户。
input, char* output)	
void GaussianNoise(char*	高斯噪声。
input, char* output, double	
sigma) void	椒盐噪声。
,	/似血·朱尸。
SaltAndPepperNoise(char* input, char* output)	
void MeanFilter(char*	<b>均</b> 店滤波
input, char* output, int	均值滤波。
filterSize)	
void GaussianFilter(char*	   高斯滤波。
input, char* output, double	PU /y  f/心 f/久。
sigma)	
void MedianFilter(char*	中值滤波。
input, char* output, int size)	I LEL1/10-1/人。
void	
voru	71 从70 担 栀 伙 爺。

L Htt: or ontMoonbilton (oborv	
EfficientMeanFilter(char*	
input, char* output, int filterSize)	
double	拉宝温差 计算图像扣似度 近回传载小图
	均方误差,计算图像相似度,返回值越小图像就找知似
MeanSquaredError(char*	像就越相似。
input1, char* input2, char*	
output)	
void GrayAVS(char*	input 是输入文件名, output 是输出文件
input, char* output, float	名。支持 8 位 BMP 图像。
k, float b)	<b>主</b> 一
void	直方图均衡化, input 是输入文件名,
HistogramEqualize24(char*	output 是输出文件名。支持 24 位 BMP 图
input, char* output)	像。
void	矩阵变换。
MatrixTransformation(char*	
input, char* output)	0.0
void Binarization(char*	二值化。
input, char* output)	
void	分离出蓝色通道。
ChannelSeparation_B(char*	
input, char* output)	
void	分离出绿色通道。
ChannelSeparation_G(char*	
input, char* output)	
void	分离出红色通道。
ChannelSeparation_R(char*	
input, char* output)	
void Inverse(char*	反转。
input, char* output)	
void	直方图均衡化。
HistogramEqualization8(char*	
input, char* output)	
void Smooth(char*	平滑。
input, char* output)	
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 名。支持 8 位 BMP 图像。 void TextTest(char* input 是输入文件名,output 是输出文件 名。支持 8 位 BMP 图像。 void RedChannel(char* input, char* output)
voidTextTest(char* input 是输入文件名, output 是输出文件 名。支持 8 位 BMP 图像。voidRedChannel(char* input, char* output)生成图像的红色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。voidGreenChannel(char* input, char* output)生成图像的绿色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。voidBlueChannel(char* input, char* output)生成图像的蓝色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。voidBlueChannel(char* input, char* output)生成图像的蓝色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。void直方图统计, input 是输入文件名, output
input, char* output)  void RedChannel (char* 生成图像的红色通道图像,input 是输入文件名,output 是输出文件名。支持 24位 BMP 图像。  void GreenChannel (char* 自put, char* output)  void BlueChannel (char* 生成图像的绿色通道图像,input 是输入文件名,output 是输出文件名。支持 24位 BMP 图像。  void BlueChannel (char* 生成图像的蓝色通道图像,input 是输入文件名,output 是输出文件名。支持 24位 BMP 图像。  void BlueChannel (char* 生成图像的蓝色通道图像,input 是输入文件名,output 是输出文件名。支持 24位 BMP 图像。
voidRedChannel (char* input, char* output)生成图像的红色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP图像。voidGreenChannel (char* input, char* output)生成图像的绿色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP图像。voidBlueChannel (char* input, char* output)生成图像的蓝色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP图像。void自方图统计, input 是输入文件名, output
input, char* output)件名,output 是输出文件名。支持 24 位 BMP 图像。void GreenChannel(char* input, char* output)生成图像的绿色通道图像,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。void BlueChannel(char* input, char* output)生成图像的蓝色通道图像,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。void input, char* output)性名,output 是输出文件名。支持 24 位 BMP 图像。void input, char* output)直方图统计,input 是输入文件名,output
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图像。 void 直方图统计,input 是输入文件名,output
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图像。 void 直方图统计,input 是输入文件名,output
voidGreenChannel(char* input, char* output)生成图像的绿色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP图像。voidBlueChannel(char* input, char* output)生成图像的蓝色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP图像。void直方图统计, input 是输入文件名, output
input, char* output)件名,output 是输出文件名。支持 24 位 BMP 图像。voidBlueChannel(char* input, char* output)生成图像的蓝色通道图像,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。void直方图统计,input 是输入文件名,output
voidBlueChannel(char* input, char* output)生成图像的蓝色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP图像。void直方图统计, input 是输入文件名, output
voidBlueChannel(char*生成图像的蓝色通道图像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP图像。void直方图统计, input 是输入文件名, output
input, char* output)件名,output 是输出文件名。支持 24 位 BMP 图像。void直方图统计,input 是输入文件名,output
图像。 void 直方图统计, input 是输入文件名, output
void 直方图统计, input 是输入文件名, output
HISTOGRAMStatistics(char*   声输出 X 件名。文存 24 W 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* 旋转, input 是输入文件名, output 是输出
input, char* output, double 文件名。degree 是旋转的度数。支持 24 位
degree) 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。

OstuThresholdSegmentationMet	件名,output 是输出文件名。支持 24 位 BMP
hod (char* input, char*	图像。
output) void Repudiation(char*	将伪彩图片反白, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 24 位 BMP 图
imput, char voutput)	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
	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	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。
<pre>Image1* LoadImage1(char* input)</pre>	BMP 图像读取, input 是输入文件名。支持 8 位和 24 位 BMP 图像。
Thiput)	返回 Imagel 型数据,Imagel 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 SaveImage1(char*	将 Imagel 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。支持 8 位和 24 位 BMP 图像。
	Image1 型数据的结构如下:
	4
	typedef struct
	{
	{    int width;
	<pre>int width; int height;</pre>
	int width; int height; int channels; //图像通道数
	int width; int height; int channels; //图像通道数 unsigned char* Data; //像素数据
void	int width; int height; int channels; //图像通道数 unsigned char* Data; //像素数据 }Imagel;
void ImageContrastExtension(char*	int width; int height; int channels; //图像通道数 unsigned char* Data; //像素数据

	-1-2-61-2-
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)	之間出入口。
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.
IMAGE Image bmp load(char*	加载 BMP 图片。
filename)	
void Image_bmp_save(char*	保存 BMP 图片。
filename, IMAGE im)	
IMAGE	缩放图片(最近邻插值法)。
TransformShapeNearest(IMAGE	4日/人区11 (東久) (中国日147)。
_	
input, unsigned int newWidth,	
unsigned int newHeight)	熔边原 县 (河外县44年)
IMAGE	缩放图片(双线性插值法)。
TransformShapeLinear(IMAGE	

input, unsigned int newWidth,	
unsigned int newHeight)	III III I I I A TO II ACCIONALI
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)	(6, 42 - 44 - 24 - 24 - 24 - 24 - 24 - 24
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	7.
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)	
	高斯滤波。
IMAGE	中  /y  1/心1/入 °

```
高斯滤波卷积核:
WavefilteringGauss (IMAGE
                             double KERNELS Wave Gauss[9] =
input, double
KERNELS Wave Gauss[9], int
a, double b)
                                 1, 2, 1,
                                 2, 4, 2,
                                 1, 2,1
IMAGE
                             低通滤波。
Wavefiltering LowPass(IMAGE
                             // 低通滤波卷积核 LP1
                             double KERNELS_Wave_LowPass_LP1[9] =
input, double* kernels)
                                 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] =
                                 1 / 10.0, 1 / 10.0, 1 / 10.0,
                                 1 / 10.0, 1 / 5.0, 1 / 10.0,
                                 1 / 10.0, 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
                             // 高通滤波卷积核 HP1
input, double* kernels)
                             double KERNELS Wave HighPass HP1[9] =
                                 -1, -1, -1,
                                 -1, 9, -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
IMAGE
                            均值滤波。
Wavefiltering_Average(IMAGE
                            // 均值滤波卷积核
input, double*
                            double KERNELS_Wave_Average[25] =
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, 1
                            差分边缘检测。
IMAGE
EdgeDetectionDifference(IMAG
                            // 差分垂直边缘检测卷积核
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] =
input,
        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
                             Laplace 边缘检测。
IMAGE
EdgeDetectionLaplace(IMAGE
                             // Laplace 边缘检测卷积核 LAP1
                             double KERNELS_Edge_Laplace_LAP1[9] =
input, double* kernels)
                                 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)
                             double
                             KERNELS Morphology Erosion cross[9] =
                                 0, 1, 0,
                                 1, 1, 1,
                                 0, 1, 0
IMAGE
                             膨胀。
MorphologyDilation(IMAGE
                             // 膨胀卷积核
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)
        FaceDetection(char*
                             人脸检测。
void
input, char*
             output, double*
KERNELS Wave Average)
         FaceDetection (IMAGE
                             人脸检测。
IMAGE
                             需引入以下结构体:
input1, IMAGE
             input2, double*
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:
                            IMAGE Image bmp load(char* filename);
                                            Image bmp save(char*
                            void
                            filename, IMAGE im);
                            参考:
                            // 用于处理
                            IMAGE
                                            input2
                            Image bmp load(inputfile);
                            // 用于保存
                            TMAGE
                                                        input2=
                            Image bmp load(inputfile);
                            input2=FaceDetection(input1, input2, KE
                            RNELS Wave Average);
                            // 保存图片
                            Image_bmp_save(outputfile, input2);
                            图像积分图。
void
IntegralDiagram(unsigned int
*input, unsigned int *output,
int width, int height)
                            图像压缩, input 是输入文件名, output 是
           Compress8(string
void
                            输出文件名。支持 8 位 BMP 图像。
input, string output)
                            图像解压, input 是输入文件名, output 是
       Decompression(string
void
input, string output)
                            输出文件名。支持 8 位 BMP 图像压缩后的结
                            果文件。
     HorizontalMirror(char*
                            水平镜像, input 是输入文件名, output 是
void
input, char* output)
                            输出文件名。支持 8 位 BMP 图像。
     MirrorVertically(char*
                            垂直镜像, input 是输入文件名, output 是
void
                            输出文件名。支持 8 位 BMP 图像。
input, char* output)
                            X镜像, input 是输入文件名, output 是输
void
           XMirroring(char*
                            出文件名。支持8位BMP图像。
input, char* output)
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。
     ImageConvolution(char*
                            图像卷积, input 是输入文件名, output 是
void
input, char*
            output, double**
                            输出文件名。Kernel 是卷积核,如 double
                                               \{\{-0, 225, -0, 225-
Kernel, int n, int m)
                            Kerne1[3][3]
                                          =
                            0.225, \{-0.225, 1, -0.225\}, \{-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。
:1 C : 1M : E: ( 1 : : : : : : : : : : : : : : : : :	
void SpatialMeanFiter(char*	空间均值过滤器,参考: radius=3。
input, char* output, int	
radius)	<b>克门上住牙连四</b>
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	
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	
* input, char* output, int	
radius, float T)	
void Mosaic(char*	图像拼接,w和h是输出图像的宽和高。支
input, char* output, int w, int	持 PNG 图像。
h)	V. 1
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)	0.10
void STDFT1(char*	参考: inv=false。
input, char* output, bool inv)	
void STDFT2(char*	参考: inv=false。
input, char* output, bool inv)	TT /ft sket I D N h N I
void SpectrumShaping(char*	图像频域滤波, FFT 变换相位谱,
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
	域填充的颜色,如 color=100。支持 8 位 BMP
	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	
threshold, BMPMat target)	

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 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)	
void SharpenUSM(char*	USM 锐化,参考: radius=5, amount=400,
input, char* output, int	threshold=50。
radius, int amount, int	
threshold)	
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	
lx, float ly)	
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)	771177
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
input, BMPMat** output, BMPMat	修复权重系数。
target, BMPMatdouble weight)	沙交权里尔奴。
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
	人小相同力,target 足行点像系,wergint 足   修复权重系数。
char** input, unsigned char**	
output, unsigned char	
target, double weight)	innut 目绘)文件名 output 目绘山文件
RotateRight90Degrees(char*	input 是输入文件名, output 是输出文件 名。支持 8 位 BMP 图像, 向右旋转 90 度。
input, char* output)	石。又持6位DMF 图像,四石灰石 90 反。
void	innut 見給)文件名 output 見給山文件
	input 是输入文件名, output 是输出文件 名。支持 8 位 BMP 图像, 向左旋转 90 度。
RotateLeft90Degrees(char*	石。又行 O 位 DMF 图像,问左旋将 90 度。
<pre>input, char* output) void</pre>	图像旋转, input 是输入文件名, output 是
,	
input, char* output, double	輸出文件名。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。 图像旋转 input 思绘》文件名 output 思
void Rotation24(char*	图像旋转, input 是输入文件名, output 是 输出文件名, 支持 24 位 PMP 图像 Anglo 具
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 red, unsigned char	围绕的中心点的坐标; red、green、blue 分别具旋转与非原图区域更填充的额色的红
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)	的角度。
void	灰度图像旋转 90。
imgRotate90Gray(unsigned	为()文国 ()()()()()()()()()()()()()()()()()()()
char *input, unsigned char	
*output, int sw, int sh, int	
*dw, int *dh)	
void	
imgRotate90Color(unsigned	TO LET BURGETY OUT
char *input, unsigned char	
*output, int sw, int sh, int	
*dw, int *dh)	
void	
imgRotate270Gray(unsigned	OCITIES DO SE LA COLOR DE LA C
char *input, unsigned char	
*output, int sw, int sh, int	
*dw, int *dh)	
void	彩色图像旋转 270。
imgRotate270Color(unsigned	
char *input, unsigned char	
*output, int sw, int sh, int	
*dw, int *dh)	
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)	
void	彩色图像旋转 180, 结果保存在原输入数组
imgRotate180Color(unsigned	中。
char *Img, int w, int h)	
void imgRBExchange(unsigned	彩色图像 R、B 互换,结果保存在原输入数
char *Img, int w, int h)	组中。
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)	
a, 110at b)	

void NoiseExp(char*	指数噪声,参考: a=0.1。
input, char* output, float a)	Jロ外(水) (*) 多つ・ローロー
void NoiseImpulse(char*	椒盐噪声,参考: a=0.2, b=0.2。
input, char* output, float	Main
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 m1, 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,
int	255, 255 } };
int MinimumValueOfImagePixels(ch	返回图像像素的最小值,filename 是输入
ar* filename)	门图像文件石。文诗 0 四种 2年 四 DMI 图像。
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)	

woid ColorTransfer1 (char* input1, char* input2, char* output, int radius, int smooth)  void OilpaintFilter (char* input, char* output, int radius, int smooth)  void HaloFilter (char* input, char* output, int ratio)  void GrayHistogram (char* input, char* output, int hWidth, int hHeight)  void RedHistogram (char* input, char* output, int hWidth, int hHeight)  void GreenHistogram (char* input, char* output, int hWidth, int hHeight)  void GreenHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int imgBit)  void BlueHistogram (char* input, char* output, int ingBit)  void Histogram Equalization (char* input, char* output)  void HaloFilter (char* input, char* output, int hHeight)  void GrayHistogram (char* input, char* output, int hHeight)  void Frank (char* input, char* input, char* output, i		
woid ColorTransferl (char* input1, char* input2, char* output)  woid OilpaintFilter (char* input, char* output, int radius, int smooth)  woid HaloFilter (char* input, char* output, int ratio)  woid GrayHistogram (char* input, char* output, int hWidth, int hHeight)  woid RedHistogram (char* input, char* output, int hWidth, int hHeight)  woid GreenHistogram (char* input, char* output, int hWidth, int hHeight)  woid GreenHistogram (char* input, char* output, int hWidth, int hHeight)  woid BlueHistogram (char* input, char* output, int hWidth, int hHeight)  woid BlueHistogram (char* input, char* output, int hWidth, int hHeight)  woid BlueHistogram (char* input, char* output, int imgBit)  woid Histogram Equalization (char* input, char* output)  woid Histogram Equalization (char* input, char* output, int hWidth, int hHeight)  woid Histogram Equalization (char* input, char* output, int hWidth, int hHeight)  woid Histogram Equalization (char* input, char* output, int hWidth, int hHeight)  woid Histogram Equalization (char* input, char* output, int hWidth, int hHeight)  woid Histogram Equalization (char* input, char* output, int hWidth, int hHeight)  woid Histogram Equalization (char* input, char* output, char* out	void HSV(char* input, char*	图像色调饱和度明度调节,参考: h=120,
input1, char* input2, char* output, in radius, int smooth)  void HaloFilter (char* input, char* output, int ratio)  void GrayHistogram(char* input, char* output, int hwidth, int hHeight)  void RedHistogram(char* input, char* output, int hwidth, int hHeight)  void GreenHistogram(char* input, char* output, int hwidth, int hHeight)  void GreenHistogram(char* input, char* output, int hwidth, int hHeight)  void BlueHistogram(char* input, char* output, int hwidth, int hHeight)  void BlueHistogram(char* input, char* output, int hwidth, int hHeight)  void BlueHistogram(char* input, char* output, int hwidth, int hHeight)  void BlueHistogram(char* input, char* output, int imgBit)  void BlueHistogram(char* input, char* input, char* output, int imgBit)  void BlueHistogram(char* input, char* input, char* inp	output, int h, int s, int v)	
woid OilpaintFilter(char* input, char* output, int radius, int smooth)  void	`	颜色转移,支持 BMP 图像。
void OilpaintFilter(char* input, char* output, int radius, int smooth) void HaloFilter(char* output, int ratio) void GrayHistogram(char* output, int hWidth, int hHeight) void RedHistogram(char* output, char* output, int hWidth, int hHeight) void GreenHistogram(char* input, char* output, int hWidth, int hHeight) void GreenHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int imgBit) void HistogramEqualization2(char* input, char* output, int imgBit) void HistogramEqualization3(char* input, char* output) void HistogramEqualization4(char* input, char* output) void HistogramEqualization4(char* input, char* output, int hWidth, int hHeight) void FablogramEqualization(char* input, char* output, int hWidth, int hHeight) void FablogramEqualization(char* input, char* output, int hWidth, int hHeight) void FablogramEqualization(char* input, char* output, int hWidth, int hHeight) void FablogramEqualization(char* input, char* output, int hWidth, int hHeight) void FablogramEqualization(char* input, char* output, int hWidth, int hHeight) void FablogramEqualization(char* output, char* output, int hWidth, int hHeight) void FablogramEqualization(char* output, char* output, int hWidth, int hHeight) void FablogramEqualization(char* output, char* output, int hWidth, int hHeight) void FablogramEqualization(char* output, char* output, int hWidth, int hHeight)	input1, char* input2, char*	
input, char* output, int radius, int smooth)  void HaloFilter(char* input, char* output, int ratio)  void GrayHistogram(char* input, char* output, int hWidth, int hHeight)  void RedHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void HistogramEqualization2(char* input, char* output, int imgBit)  void La ja	output)	
radius, int smooth)  void HaloFilter (char* input, char* output, int ratio)  void GrayHistogram (char* input, char* output, int hWidth, int hHeight)  void RedHistogram (char* input, char* output, int hWidth, int hHeight)  void GreenHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int hWidth, int hHeight)  void BlueHistogram (char* input, char* output, int ingBit)  void BlueHistogram (char* input, char* output, int ingBit)  void BlueHistogram (char* input, char* output, int ingBit)  void En	void OilpaintFilter(char*	油画滤镜,参考: radius=10, smooth=100。
void HaloFilter(char* input, char* output, int ratio)  void GrayHistogram(char* input, char* output, int hWidth, int hHeight)  void RedHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int imput, char* output)  void HistogramEqualization3(char* input, char* output)  void HistogramEqualization4(char* input, char* output)  void HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hHeight=100。  Earlie Farkition(char* input, cha	input, char* output, int	
void HaloFilter(char* input, char* output, int ratio)  void GrayHistogram(char* input, char* output, int hWidth, int hHeight)  void RedHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int imput, char* output)  void HistogramEqualization3(char* input, char* output)  void HistogramEqualization4(char* input, char* output)  void HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hWidth, int hHeight)  void FarkitiogramEqualization(char* input, char* output, int hHeight=100。  Earlie Farkition(char* input, cha	radius, int smooth)	
input, char* output, int ratio)  void GrayHistogram(char* input, char* output, int hWidth, int hHeight)  void RedHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int imgBit)  void BlueHistogram(char* input, char* output)  properties a from the input from the imgBit	void HaloFilter(char*	晕角滤镜,参考: ratio=100。
ratio) void GrayHistogram(char* input, char* output, int hWidth, int hHeight) void RedHistogram(char* input, char* output, int hWidth, int hHeight) void GreenHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int imgBit)  void		
yoid GrayHistogram(char* input, char* output, int hWidth, int hHeight)  void RedHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void HistogramEqualization2(char* input, char* output, int imgBit)  void BlueHistogram(char* input, char* output, int imgBit)  void Englia for Mark in the fight input in the fight input, char* output, int ing fight input, char* output)  void HistogramEqualization4(char* input, char* output)  void HistogramEqualization4(char* input, char* output, int hWidth, int hHeight)  void Englia for Mark input be fight in		
input, char* output, int hWidth, int hHeight)  void RedHistogram(char* input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int imgBit)  void HistogramEqualization2(char* input, char* output, int imgBit)  void BlueHistogram(char* input, char* output, int imgBit)  void BlueHistogram(char* input, char* output, int imgBit)  void BlueHistogram(char* input, char* output, int imgBit)  void Bhy Bqc imgBit 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。专持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。专持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。专持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。专持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。专持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。专持 8 位和 24 位 BMP Bqc input 是输入文件名, output 是输出文件名。专持 8 位和 24 位 BMP Bqc input 是输入文件名称, out 是输出文件名称。  void Fractional Equalization(char* input, char* output, inthe i		灰度 直方图, 参考: hWidth=256,
width, int hHeight) void RedHistogram(char* input, char* output, int hWidth, int hHeight) void GreenHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int imgBit) void		
void RedHistogram(char* input, char* output, int hWidth, int hHeight) void GreenHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int imgBit) void		
input, char* output, int hWidth, int hHeight)  void GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BineHistogram(char* input, char* output, int imgBit)  void Ej By by K, input 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP 图像。imgBit 是输入图像的位数。  input, char* output)  void Ej By by K, input 是输入文件名, output 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP 图像。imgBit 是输入图像的位数。  input, char* output)  void Ej By by K, input 是输入文件名, output 是输入文件名和。  void Ej By By K, input 是输入文件名和, output 是输出文件名。支持 8 位和 24 位 BMP 图像。input 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP 图像。input 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP 图像。input 是输入文件名, output 是输出文件名。 支持 8 位和 24 位 BMP 图像。input 是输入文件名, output 是输入 文件名, output 是输入 output 是输入 文件名, output 是输入 文件名, output 是输入 output		红色通道直方图, 参考: hWidth=256.
woid GreenHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void Eja By 例化,input 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP 图像。imgBit 是输入图像的位数。  input, char* output, int imgBit)  void Eja By 例化,input 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP 图像。imgBit 是输入图像的位数。  input, char* output)  void Eja By 例化,input 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP 图像。 input, char* output)  void Eja By 例化,input 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP 图像。input 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP 图像。input 是输入文件名,output 是输出文件名称,out 是输出文件名称。  input, char* output, int hWidth, int hHeight)  void		-
woid GreenHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int hWidth, int hHeight) void BlueHistogram(char* input, char* output, int imgBit)  void		
input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void		绿色诵道直方图、参老·hWidth=956
Nwidth, int hHeight   woid   BlueHistogram(char* input, char* output, int hWidth, int hHeight   woid   直方图均衡化,input 是输入文件名,output, ehar* output, int imgBit   imput, char* output   int imgBit   imput, char* output   int input, char* output   input, char* output, inthwidth, int hHeight   input, char* output, inthwidth, inthwidthwidthwidthwidthwidthwidthwidthwid		
woid BlueHistogram(char* input, char* output, int hWidth, int hHeight)  void	1	inicigit 100°
input, char* output, int hWidth, int hHeight)  void  proid  input, char* output, int ingut, char* output, int imgBit)  void  HistogramEqualization2(char* input, char* output, int imgBit)  void  HistogramEqualization3(char* input, char* output)  void  HistogramEqualization4(char* input, char* output)  void  HistogramEqualization4(char* input, char* output)  void  HistogramEqualization(char* input, char* output)  void  HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void  GrayHistogramEqualization(ch  FrayHistogramEqualization(ch  MHeight=100。  hHeight=100。  a p B 均衡化,input 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP 图像。input 是输入文件名标,out 是输出文件名称。  a p B 均衡化,参考:hWidth=256,hHeight=100。  亦度直方图,参考:hWidth=256,hHeight=100。		族 岳 通 道 吉 方 図
hWidth, int hHeight) void listogramEqualization2(char* input, char* output, int imgBit)  void listogramEqualization3(char* input, char* output)  void listogramEqualization3(char* input, char* output)  void HistogramEqualization4(char* input, char* output)  void HistogramEqualization4(char* input, char* output)  void HistogramEqualization(char* input, char* output)  void HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void  frayHistogramEqualization(ch frayHistogramEqualization		
Description		imeignt-100°
HistogramEqualization2(char* input, char* output, int imgBit)  void  HistogramEqualization3(char* input, char* output)  void  HistogramEqualization3(char* input, char* output)  void  HistogramEqualization4(char* input, char* output)  void  HistogramEqualization4(char* input, char* output)  void  HistogramEqualization(char* input, char* output)  void  Figure BMP B(R) imput 是输入文件名,output 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP B(R) input 是输入文件名称,out 是输出文件名称。  void  HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void  Figure BMP B(R) imgBit 是输入文件名,output 是输入文件名,output 是输出文件名。支持 8 位和 24 位 BMP B(R) input 是输入文件名称,out 是输出文件名称。  Imput, char* output, int hHeight by		古主图均衡从 input 見給)立併夕
input, char* output, int imgBit)  woid  HistogramEqualization3(char* input, char* output)  woid  HistogramEqualization4(char* input, char* output)  woid  HistogramEqualization4(char* input, char* output)  woid  HistogramEqualization4(char* input, char* output)  woid  HistogramEqualization(char* input, char* output)  woid  HistogramEqualization(char* input, char* output, inthwidth, int hHeight)  woid  GrayHistogramEqualization(char* input, char* output, inthwidth, int hHeight)  woid  From Equalization (char* input, char* output, inthwidth, int hHeight)  From Equalization (char* input, char* output, inthwidth, int hHeight)  From Equalization (char* input, char* output, inthwidth, int hHeight)  From Equalization (char* input, char* output, inthwidth, int hHeight)  From Equalization (char* input, char* output, inthwidth, inthymidth, inth		
imgBit)  void  BistogramEqualization3(char* output, char* output)  void  HistogramEqualization4(char* input, char* output)  binput, char* output)  void  HistogramEqualization4(char* input, char* output)  void  HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void  frayHistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void  frayHistogramEqualization(char* input, char* output, int hWidth, int hHeight)  x 度 直 方 图 ,参考: hWidth=256, hHeight=100。		
直方图均衡化,input 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP 图像。   void		BMP 图像。1mgblt 定制八图像的位数。
HistogramEqualization3(char* input, char* output)  void  woid HistogramEqualization4(char* input, char* output)  woid input, char* output)  woid  woid  woid  Woid  Woid  Woid  FirstogramEqualization(char* input, char* output)  woid  Width, int hHeight)  woid  GrayHistogramEqualization(char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)  woid  Frage American Strain (char* input, char* output, int hWidth, int hHeight)		<b>本之圆均炼</b> 从
input, char* output)  void  HistogramEqualization4(char* input, char* output)  void  Woid  HistogramEqualization(char* input, char* output)  woid  HistogramEqualization(char* input, char* output, inthwidth, int hHeight)  void  GrayHistogramEqualization(char* input, char* output, inthwidth, int hHeight)  woid  GrayHistogramEqualization(char* input, char* output, inthwidth, int hHeight)  woid  Free E F B, \$\frac{\pi}{2}\$, \$\frac{\pi}{2}\$; hWidth=256, hHeight=100.		
roid HistogramEqualization4(char* input, char* output)  void HistogramEqualization4(char* input, char* output)  void HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void  GrayHistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void  Free E 方图均衡化,参考:hWidth=256,hHeight=100。  Free E 方图,参考:hWidth=256,hHeight=100。		
HistogramEqualization4(char* input, char* output)  void HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void GrayHistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void GrayHistogramEqualization(char* input, char* output, int hWidth, int hHeight)  x 度 直 方 图 , 参 考: hWidth=256 , hHeight=100。		1 1 2 2
input, char* output)  BMP 图像。input 是输入文件名称,out 是输出文件名称。  void  直方图均衡化,参考:hWidth=256, hHeight=100。  input, char* output, int hWidth, int hHeight)  void  GrayHistogramEqualization(ch hHeight=100。		
woid a 方图均衡化,参考: hWidth=256, hHistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void		
void 直方图均衡化,参考: hWidth=256, HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void 灰度直方图,参考: hWidth=256, HHeight=100。	input, char* output)	
HistogramEqualization(char* input, char* output, int hWidth, int hHeight)  void 灰度直方图,参考: hWidth=256, hHeight=100。		
input, char* output, int hWidth, int hHeight)  void 灰度直方图,参考: hWidth=256, GrayHistogramEqualization(ch hHeight=100。		_
hWidth, int hHeight) void 灰度直方图,参考: hWidth=256, GrayHistogramEqualization(ch hHeight=100。	HistogramEqualization(char*	hHeight=100。
void 灰度直方图,参考: hWidth=256, GrayHistogramEqualization(ch hHeight=100。	I - · · ·	
GrayHistogramEqualization(ch hHeight=100。	hWidth, int hHeight)	
	void	灰度直方图, 参考: hWidth=256,
ar* input, char* output, int	GrayHistogramEqualization(ch	hHeight=100。
	ar* input, char* output, int	
hWidth, int hHeight)	hWidth, int hHeight)	
void 红色通道直方图,参考: hWidth=256,	void	好 毎 掻 古 古 図   <del>会 孝</del>   LW; J+b=956
RedHistogramEqualization(cha   hHeight=100。		红色通过且刀图,参考: IIWIUII-250,

r* input, char* output, int	
hWidth, int hHeight)	
void	绿色通道直方图,参考: hWidth=256,
GreenHistogramEqualization(c	hHeight=100。
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	
hWidth, int hHeight)	
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)	# A Z W + A E
void	蓝色通道直方图,参考: hWidth=256,
BlueHistagramStretch(char*	hHeight=100。
input, char* output, int	
hWidth, int hHeight)	中体演说: 4 目於) 文件名 4 4 目
void MedianFiltering1(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。 阈值处理, input 是输入文件名, output 是
ThresholdProcessing(char*	阈值处理, input 定制入文件名, output 定   输出文件名。支持 8 位 BMP 图像。Threshold
input, char* output, int	是阈值相关参数,如 Threshold=0.001。
Threshold)	た
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)	
void ToRIM(char* input, char*	一般图像转到 RIM 图像,支持 PNG、JPG 和
output)	TGA 图像。
· · · · · · · · · · · · · · · · · ·	¬ W**

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*	机的位图打印输出,支持的热敏打印机的位
input, char* output, ARRAY3	图打印指令为〈strong〉ESC *〈/strong〉指
skip_cmd, unsigned short	令。
PRINTER_TYPE_BMP, unsigned	typedef unsigned char ARRAY3[3];参考:
char mode, unsigned int	output="output.pbin", skip cmd =
FILE TYPE AD, unsigned char	{0x1B, 0x4A, 0x00}, PRINTER TYPE BMP 是
a, unsigned char b)	打印机位图打印指令码标识,
	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.
void WhiteBalance(const	白平衡。
char* input, const char*	
output)	
void Sobel(char* input, char*	Sobel 算子, magnScale=0.35 ,
output, double	threshold=130。支持 PGM 和 PBM 图像。
magnScale, double threshold)	threshold 100° 文刊 10m 和 1Dm 国家。
void Canny(char* input, char*	Canny 算 子 , magnScale=0.35 ,
output, double	lowThreshold=55, highThreshold=120。支
magnScale, double	持 PGM 和 PBM 图像。
lowThreshold, double	14 LOW 11 L DW 12 19/0
highThreshold)	
void BlackWhite(char*	黑白化,threshold=100,background=0。
input, char* output, int	支持 PGM 和 PBM 图像。
threshold, int background)	XIII OM III EI IX
void	区域连通,threshold=100,background=0,
ConnectedComponents(char*	threshold1=100。支持 PGM 和 PBM 图像。
input, char* output, int	200 × 10 10 10 10 10 10 10 10 10 10 10 10 10
threshold, int background, int	
threshold)	
void CleanImage(char*	清洁图像。支持 PGM 和 PBM 图像。
input, char* output)	11.12 H M. 2011 1 2011 H M.
void NoiseImage(char*	噪声化图像,probability=0.1。支持 PGM
input, char* output, float	和 PBM 图像。
probability)	
void	圆检测。scale1=1,gamma1=1.0,
HoughTransformCircle1(char*	magnScale=0.5 , lowThreshold=85 ,
input, char* output, double	highThreshold=150, scale=0, gamma=1.0,
sigma, int kernelSize, int	sigma 和 kernelSize 用于平滑的高斯 5x5
scale, double gamma, double	内核, sigma=1.0, kernelSize=5。
, , , , , , , , , , , , , , , , , , , ,	, , , , , , , , , , , , , , , , , , ,

magnScale, double
lowThreshold, double
highThreshold, int scale1,
double gamma1)

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

HoughTransformCircle2(char\*
input, char\* output, int
number, int minDist, double
sigma, int kernelSize, int
scale, double gamma, double
magnScale, double
lowThreshold, double
highThreshold, int scale1,
double gamma1)

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,number=10 表示图像的目视检查有 10 个圆圈,minDist=35。

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。

double\*\*

HoughTransformCircle3(char\*
input, char\* output, int
number, int minDist, double
sigma, int kernelSize, int
scale, double gamma, double
nagnScale, double
lowThreshold, double
highThreshold, int scale1,
double gamma1)

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, number=10 表示图像的目视检查有 10 个圆圈, minDist=35。

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。

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。

void	形状边缘检测, CANNY THRESH4=35,
ShapeEdgeDetection1(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 THRESH4=35,
ShapeEdgeDetection2(char*	CANNY_blur4=7。支持 PNG 图像。
input, char* output, unsigned	CHIMI_BILLIT 18 XII INO ELIKE
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	OHMI_DBOK 120 X14 THO ELIKO
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	ommi_spec 120 X14 The Elike
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)	
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)	
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	
CANINIV 1.1 2)	
CANNY_b1ur3)	
void	形状边缘检测,CANNY_THRESH3=45,
	形状边缘检测,CANNY_THRESH3=45, CANNY_blur3=10。支持PNG图像。
void	_
void ShapeEdgeDetection8(char*	_
void ShapeEdgeDetection8(char* input, char* output, unsigned	_

## 其他处理

void E	ncode(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	<u>;</u> )		输入文件名,output 是输出文件名。
void FileCompress(char *input ,		ar *input ,	文件压缩, input 是输入文件名,
char *output)			output 是输出文件名。
void	FileDecomp	ression(char	文件压缩结果解压缩, input 是输入
*input	t, char *output	)	文件名,output 是输出文件名。

## 高级算子

void BlobAnalysis(char*	Blob 分析, c1 和 c2 是与颜色相关的
input, char* output, int c1, int c2)	参数,参考: c1=128, c2=127。支持 BMP 图像。
void BlobAnalysis1(char*	Blob 分析, c1 和 c2 是与颜色相关的
input, char* output, int c1, int c2)	参数,参考: c1=128, c2=127。支持 BMP
	图像。
void	验证码生成。
VerificationCodeGeneration(char*	sigma=10, noise_type=2, a=10, b1=12
inputText, char* output, int	8, b2=127, b3=2, b4=8, b5=12, b6=0,
num, int foint, int a, int bl, int	b7=1, b8=-100, b9=-100, b10=1,
b2, int b3, int b4, int b5, int b6, int	b11=3, b12=6, int b13=40,
b7, int b8, int b9, int b10, int	foint=30, num 是 字 符 数
b11, int b12, int b13, double	量,depth=1,spectrum=3,shared=0
sigma, unsigned int noise_type, int	
width, int height, int depth, int	
spectrum, bool shared)	# HAND 1 1 1 1 10000 1 0 00
void CornerDetection(char*	角点检测, threshold=10000, k=0.06,
input, char* output, float	sigma=1.0, width=640, height=480,
threshold, float k, float	channels=1。支持 PNM 图像。
sigma, int width, int height, int	
<pre>channels) vector<keypoint></keypoint></pre>	
CornerDetection1(char*	用 点 極 例 ,
input, char* output, float	sigma=1.0, width=640, height=480,
threshold, float k, float	channels=1。支持 PNM 图像。
sigma, int width, int height, int	需引入以下结构体
channels)	typedef struct {
,	float x;
	float y;
	float score;
	} Keypoint;
vector <corner::keypoint></corner::keypoint>	角点检测,返回角点数据。
CornerDetection(char* input, int	threshold=2000,k=1,sigma=1.2。支
width, int height, int	持 PNM 图像。
channels, float threshold, float	需引入以下命名空间:
k, float sigma)	namespace Corner {

```
struct Keypoint {
                                         float x:
                                         float y;
                                         float score;
                                  };
                                  特征归一化统计,参考: sigma=2。支
void Structure(char* input, char*
output, float sigma)
                                  持多种图像格式。
                                  角点检测,参考: sigma=2, method=0。
void Cornerness(char* input, char*
                                  支持多种图像格式。
output, float sigma, int method)
void Corners(char*
                                  角点检测, 参考: sigma=2,
                     input, char*
output, float sigma, float thresh,
                                  thresh=0.4, window=5, nms=3,
                                  corner method=0。支持多种图像格
      window.
int
                int
                      nms.
                             int
corner method)
                                  式。
void FindMatch(char* input1, char*
                                  特征匹配,参考: thresh3=5, k=10000,
input2, char*
                    output, float
                                  cutoff=50, thresh4=5, sigma=2,
thresh3, int k, int cutoff, float
                                  thresh=0.4, window=5, nms=3,
thresh4, float
                 sigma,
                                  corner method=0 , sigma1=2 ,
                           float
thresh, int window, int nms, int
                                  thresh1=0.4, window1=5, nms1=3,
corner method, float sigmal, float
                                  corner method1=0 , sigma2=2 ,
thresh1, int window1, int nms1,
                                  thresh2=0.4, window2=5, nms2=3,
int corner method1, float sigma2,
                                  corner method2=0 , sigma5=2 ,
                                  thresh5=0.4, window5=7, nms5=3,
float thresh2, int window2,
                                  corner_method5=0 , sigma6=2 ,
nms2,
       int
           corner_method2, float
sigma5,
          float
                  thresh5.
                             int
                                  corner method6=0, thresh6=0.3,
                                  window6=7
                                                     nms6=3
window5,
            int
                   nms5,
                             int
corner method5, float sigma6,
                                  inlier thresh6=5, iters6=1000,
                             int
                                  cutoff6=50, acoeff6=0.5。支持多种
corner method6,
                 float
                        thresh6,
int window6,
               int
                    nms6,
                                  图像格式。
                           float
inlier thresh6, int iters6,
                            int
cutoff6, float acoeff6)
vector (Descriptor)
                                  角点检测,返回检测结果。参考:
HarrisCorner (char*
                                  sigma1=2, thresh1=0.4, window1=5,
                     input, char*
output, float
                sigmal,
                           float
                                  nms1=3, corner method1=0。支持多种
thresh1, int window1, int nms1,
                                  图像格式。
int corner method1)
                                  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) {}
                                  描述匹配项,返回描述结果。参考:
vector < Match >
                                  sigma1=2, thresh1=0.4, window1=5,
MatchDescriptors(char*
input1, char*
                     input2, char*
                                  nms1=3 , corner_method1=0 ,
output, float
                sigmal,
                           float
                                  sigma2=2, thresh2=0.4, window2=5,
threshl, int windowl, int nmsl,
                                  nms2=3, corner method2=0。支持多种
int corner method1, float sigma2,
                                  图像格式。
                                  需引入以下结构体:
float thresh2, int window2, int
                                  struct Point {
nms2, int corner_method2)
                                    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) {}
                                  };
                                  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>
```

void DrawInliers (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 threshl, int windowl, int nms1, int corner method1, float float sigma2, thresh2, int window2, int nms2, int corner method2, float sigma5. float thresh5, int window5, corner method5, float nms5, int sigma6, int corner method6, float thresh6, int window6, int nms6, float inlier\_thresh6, int iters6, int cutoff6, float acoeff6)

绘制角点,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=3inlier 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 thresh1, int window1, nms1. int corner method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, corner\_method5, float nms5. int sigma6, int corner\_method6, float thresh6, int window6, int nms6, float inlier thresh6, int iters6, int cutoff6, float acoeff6)

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=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=3inlier thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5。支持多种 图像格式。

void Cylindrical (char\* input1, char\* input2, char\* f1, float output, 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 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=0.4, window5=7, nms5=3, corner\_method5=0,

nms2, corner method2, float int sigma5, float thresh5, int window5, int nms5, int corner method5, float sigma6, int corner method6, float thresh6, window6, int nms6, float int inlier thresh6, int iters6, int cutoff6, float acoeff6, float sigma7, int corner method7, float thresh7, int window7, int nms7, float inlier thresh7, int iters7, int cutoff7, float acoeff7)

sigma6=2 , corner\_method6=0 , thresh6=0.3, window6=7, nms6=3, inlier\_thresh6=5 , iters6=1000 , cutoff6=50, acoeff6=0.5, sigma7=2, corner\_method7=0 , thresh7=0.3 , window7=7 , nms7=3 , inlier\_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, 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, int 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=0.4, 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, 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 Canny(char\* input, char\*
output, int lowThreshold, int
highThreshold)

Canny 算子,至少支持 JPG 图像,input 是输入文件名,output 是输出文件名, 参考: lowThreshold=50, highThreshold=150。

void Canny(string input, string
output)

Canny 算子,参考: output="output"。 支持 BMP 文件。

void Canny(string input, char\*
output, float sigma, float
threshold)

Canny 算子,参考: sigma=6.0, threshold=3.5。支持BMP文件。

void Hough (char\* input, char\*

霍夫变换, 参考: sigma=6.0,

output, float sigma, float threshold, double thre_val, unsigned char* color)	threshold=3.5, thre_val=0.5, color 用于设置画出的标定点和线的颜色, 如: color[3]={0,0,255}。支持 BMP 文 件。
<pre>void KMeans(string input, unsigned int Clusters, char* output)</pre>	K-Means 聚类,input 是输入文件名, Clusters 是聚类的种类数目,output 是输出文件名。
<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。
void DES_Decrypt(char *CipherFile, char *Key,char *PlainFile)	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。
<pre>void HoughTransform(char* input, char* output, unsigned char threshold)</pre>	霍夫变换, input 是输入的 RAW 文件, output 是输出的 RAS 文件, threshold=100。
static void EdgeDetectionWithoutNonmaximum(co nst LPCTSTR input, const LPCTSTR output, double a, double b, double c)	边缘检测,参考: a=0.33, b=0.33, c=0.33。支持 24 位 BMP 图像。
static void CannyEdgeDetection(const LPCTSTR input, const LPCTSTR output, double a, double b, double c, int orank, int oranb)	边缘检测,参考: orank=20, oranb=80。 支持 24 位 BMP 图像。
static void HoughTransform(const LPCTSTR input, const LPCTSTR output, double a, double b, double c, int orank, int oranb)	霍夫变换,参考: a=0.33, b=0.33, c=0.33, orank=20, oranb=80。支持 24 位 BMP 图像。
void BoxBlurBasic(string input, string output)	基础方框模糊,支持 PNG 文件。
<pre>void PGMSobel(char* input, char* output, int</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 文件。 参考模板: 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}} int max = -9999 int min = 9999

void PGMSobelX(char\* input, char\* X 方向滤波, input 是输入文件名, output 是输出文件名。支持 P5 格式 output, int Mx[3][3], int 的 PGM 文件。 My[3][3], int max, int min) 参考模板: int  $Mx[3][3] = \{\{-1, 0, 1\}, \{-2, \}\}$  $0, 2\}, \{-1, 0, 1\}\}$ int  $My[3][3] = \{\{-1, -2, -1\}, \{0, \}\}$ 0, 0,  $\{1, 2, 1\}$ int max = -9999int min = 9999void PGMSobelY(char\* input, char\* Y 方向滤波, input 是输入文件名, output, int output 是输出文件名。支持 P5 格式 Mx[3][3], int My[3][3], int max, int min) 的 PGM 文件。 参考模板: int  $Mx[3][3] = \{\{-1, 0, 1\}, \{-2, \}\}$  $0, 2\}, \{-1, 0, 1\}$ int  $My[3][3] = \{\{-1, -2, -1\}, \{0, \}\}$ 0, 0,  $\{1, 2, 1\}$ int max = -9999int min = 9999Sobel 算子, input 是输入文件名, void PGMSobel1(char\* input, char\* output, int min, int max, int output 是输出文件名。min 和 max 是 mx[3][3], int my[3][3]) 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 和 Y 方向模板。支持 P2 和 P5 格式的 PGM。 参考模板:  $int mx[3][3] = {$  $\{-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, void PGMSobelX1(char\* input, char\* output, int min, int max, int output 是输出文件名。min 和 max 是 mx[3][3], int my[3][3]) 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 和 Y 方向模板。支持 P2 和 P5 格式的 PGM。 参考模板:

```
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\}
                                                                                                                       };
void PGMSobelY1(char* input, char*
                                                                                                            Y 方向梯度, input 是输入文件名,
output, int
                                                                                                            output 是输出文件名。min 和 max 是
                                            min, int
                                                                                 max, int
mx[3][3], int my[3][3])
                                                                                                            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 和 Y 方向模板。支持
                                                                                                            P2 和 P5 格式的 PGM。
                                                                                                            参考模板:
                                                                                                            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\}
void PGMSobel2(char* input, char*
                                                                                                            Sobel 算子, input 是输入文件名,
                                                                                                            output 是输出文件名。支持 P5 格式
XOutput, char*
                                                              YOutput, char*
SobelOutput, int sobel x[3][3], int
                                                                                                            的 PGM 图像。XOutput 是输出的 X 方
sobel y[3][3], int min, int max)
                                                                                                            向的梯度图像,YOutput 是输出的Y方
                                                                                                            向的梯度图像,SobelOutput 是输出的
                                                                                                            整幅图像的 Sobel 算子计算结果, min
                                                                                                            和 max 是图像归一化的相关参数,如
                                                                                                            min=100, max=0.
                                                                                                            参考模板:
                                                                                                            int sobel x[3][3] = \{\{-1, 0, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1, 1\}, \{-1,
                                                                                                            2, 0, 2, \{-1, 0, 1\};
                                                                                                            int sobel_y[3][3]=\{\{1, 2, 1\}, \{0, 1\}\}
                                                                                                            0, 0\}, \{-1, -2, -1\}\};
                       Sobel (char*
                                                                                                            Sobel 算子, input 是输入文件名,
void
                                                                    input, char*
                                                                                                            output 是输出文件名。支持 PGM 文件。
output)
void Laplatian (char* input, char*
                                                                                                           Laplatian 算子, input 是输入文件
output)
                                                                                                            名, output 是输出文件名。支持 PGM
```

	<i>→</i> #
	文件。
void HorizSobel(char* input, char*	水平 Sobel 算子, input 是输入文件
output)	名, output 是输出文件名。支持 P5 格
	式的 PGM 图像。
<pre>void VertSobel(char* input, char*</pre>	垂直 Sobel 算子, input 是输入文件
output)	名, output 是输出文件名。支持 P5 格
• ,	式的 PGM 图像。
void PGMSobell(char* input, char*	Sobel 算子, input 是输入文件名,
output, int threshold)	output 是输出文件名。支持 P5 格式
output, int threshord)	
	的PGM图像。threshold是目标阈值,
	如 threshold=80。
void YFiltering(char* input, char*	Y 方向滤波, input 是输入文件名,
output, int sobel_x[3][3], int	output 是输出文件名。支持 P5 格式
$sobel_y[3][3]$	的 PGM 图像。
	参考模板:
	int sobel_x[3][3] = { { 1, 0,
	$\begin{bmatrix} -1 \end{bmatrix}$ ,
	{ 2,
	[0, -2],
	{ 1,
	$0, -1\}$ ;
	int sobel_y[3][3] = { { 1,
	2, 1},
	{ 0,
	0, 0},
	{-1,
	$-2, -1\}$ ;
void XFiltering(char* input,char*	X 方向滤波, input 是输入文件名,
output, int sobel_x[3][3], int	output 是输出文件名。支持 P5 格式
sobe1_y[3][3])	的 PGM 图像。
_	参考模板:
	int sobel_x[3][3] = { { 1, 0,
	-1},
	{ 2,
	[0, -2],
	{ 1,
	[0, -1];
	int sobel_y[3][3] = { { 1,
	2, 1},
	{ 0,
	0, 0},
	{-1,
	$-2, -1\}\};$
void SobelFiltering(char*	Sobel 算子, input 是输入文件名,

<pre>input, char* output, int sobel_x[3][3], int sobel_y[3][3])</pre>	output 是输出文件名。支持 P5 格式 的 PGM 图像。 参考模板: int sobel_x[3][3] = { { 1, 0, -1},
	{-1,
void PrewittFiltering(char* input, char* output, int prewitt_x[3][3], int	-2, -1}};   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。
<pre>prewitt_y[3][3])</pre>	参考模板: int prewitt_x[3][3] = { { 5, 5, 5},
	$\{ -3, \ 0, -3\}, $
	-3, -3}}; int prewitt_y[3][3] = { { 5,
	$\begin{bmatrix} -3, & -3 \}, \\ 0, & -3 \}, \end{bmatrix}$
	{5, -
<pre>void LaplacianFiltering(char* input, char* output, int laplacian[3][3])</pre>	3, -3}}; Laplace 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。laplacian 是 Laplacian 算子模板。 参考模板:
	int laplacian[3][3] = { { 1, 1, 1, 1},
	\[ \begin{aligned} \ \ -8, & 1 \end{aligned}, \]
	{ 1,   1}};
void RAWSobelEdge(char* input, char* output, int ROWS, int	Sobel 算子, input 是输入文件名, output 是输出文件名。ROWS 是图像的

COLS, int M, float	行, COLS 是图像的列, M 是滤波相关
sobelX[3][3], float sobelY[3][3])	参数,如 M=1。支持 RAW 图像。
	参考模板:
	float sobelX[3][3] = {{-1,0,1}, {-
	2, 0, 2},
	1, 0, 1}};
	float sobelY[3][3] = {{-1,-2,-1},
	{0,0,0},
	$\{1, 2, 1\}\};$
void RAWPlaceHolder(char*	边缘检测, input 是输入文件名,
input, char* output, int ROWS, int	output 是输出文件名。ROWS 是图像的
COLS, int M, float mask[3][3])	行, COLS 是图像的列, M 是滤波相关
	参数,如 M=1。支持 RAW 图像。
	参考模板:
	float mask[3][3] = $\{\{-1, -2, -1\},\$
	{0,0,0}, {1,2,1}};
void	拉普拉斯锐化滤波器, input 是输入
RAWLaplacialSharpeningFilter(char	文件名,output 是输出文件名。ROWS
* input, char* output, int ROWS, int	是图像的行大小,COLS 是图像的列大
COLS, int M, float w, float	小,M和w是滤波相关参数,如M=1,
mask[3][3])	w=1; mask 是滤波器模板。支持 RAW 图
	像。 参考模板 <b>:</b>
	多有模拟:   float mask[3][3] = {{0,1,0},
	$\{1, -4, 1\},\$
	{0,1,0}};
void	拉普拉斯算子增强, input1 是输入的
RawLaplacianEnhancement(char*	RAW 图像文件名,output1 是输出的
input1, char* output1, int	RAW 图像文件名, width 是输入图像的
width, int height)	宽, height 是输入图像的高。支持 RAW
	图像。
void SobelOperation1(char*	Sobel 算子。支持 RAW 图像。
input, char* output, int width, int	
height)	
void SobelOperation2(char*	Sobel 算子。支持 RAW 图像。
input, char* output, int width, int	
height)	D 1
void Roberts(unsigned char**	Roberts 算子,input 是输入数据,

innut ungingal abaratuk autmut)	out nut 目於山粉琨
input, unsigned char** output)	output 是输出数据。
void Roberts (BMPMat**	Roberts 算子, input 是输入数据,
input, BMPMat** output)	output 是输出数据。
void STLSection(char* input, char*	STL 切片, input 是输入的 STL 文件,
output, int sliceAmount, int	output 是输出的切片文件前缀名,
resolution, int c)	sliceAmount 是切片量,如:
	sliceAmount=50, resolution 是分辨
	率,如:resolution=260,c是执行的
	相关参数,如: c=5。
void SURF(char* input1, char*	SURF 算子,input1 和 input2 是输入
input2, char* output)	文件名,output 是输出文件名。支持
	BMP 图像。
void SobelOperator(char*	Sobel 算子,耗时较长,input 是输入
input, char* output)	文件名,output 是输出文件名。支持
	24 位 BMP 图像。
SobelImage** SobelOperator(char*	返回处理后的各像素点的坐标和对应
input)	的像素值,若是边缘点则对应白色,
	否则对应黑色。支持 24 位 BMP 图像。
	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 EdgeDetection(char*	边缘检测, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 4 位 BMP
	图像。
void EdgeDetection1(char*	边缘检测, input 是输入文件名,
input, char* output, short	output 是输出文件名。支持 8 位 BMP
sharpen[3][3])	图像。
-F	参考模板:
	short sharpen[3][3] = $\{\{1, 1, 1\},$
	{1, -8,
	1},
	$\{1, \dots, 1, \dots\}$
	1}};
void EdgeDetection2(char*	边缘检测, input 是输入文件名,
input, char* output, int a)	output 是输出文件名。a 是用于设置
input, chai. output, int a/	图像像素的相关参数,如 a=3。支持 24
	位 BMP 图像。
void EdgeDetection3(char*	边缘检测,input 是输入文件名,
input, char* output, int a)	output 是输出文件名。a 是用于设置
input, chair output, int a)	output 是他田文下石。4 是用 1 以且

	图像像素的相关参数,如 a=3。支持 24 位 BMP 图像。
void EdgeDetection4(char*	边缘检测, input 是输入文件名,
input, char* output, int a)	output 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 Roberts(char* input, char*	Roberts 边缘检测。支持 BMP 图像。
output)	
void Prewitt(char* input,char*	Prewitt 边缘检测。支持 BMP 图像。
output)	
void Sobel(char* input, char*	Sobel 边缘检测。支持 BMP 图像。
output)	
void Laplace(char* input, char*	Laplace 边缘检测。支持 BMP 图像。
output)	
void BoxBlurAdvanced(string	高级方框模糊,参考: radius=5。支持
input, string output, int radius)	PNG 文件。
void LaplacianEnhancement(char*	拉普拉斯图像增强, input 是输入文
input, char* output, int N, int	件名,output 是输出文件名。如: N=1。
LaplMask[3][3])	支持8位BMP图像。
	参考模板:
	int LaplMask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
:1 I 1 C 41/1	大並长虹亚海 : 日於入文件名
void LaplaceSmooth(char*	拉普拉斯平滑,input 是输入文件名,
input, char* output, int N, int Lap1Mask[3][3])	output 是输出文件名。如: N=1。支持   8 位 BMP 图像。
Lapimask[3][3]/	
	参考模板:   int Lap1Mask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void Sobell(char* input, char*	Sobel 算子, input 是输入文件名,
output, int N, int	output 是输出文件名。如: N=1。支持
Sb1Mask1[3][3], int	8位BMP图像。
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*	
input, char* output, int N, int	
SblMask1[3][3], int	8位BMP图像。
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* input, char*	图像倍增化, input 是输入文件名,
output, int N, int	
Sb1Mask1[3][3], int	8位BMP图像。
Sb1Mask2[3][3], int	参考模板:
Lap1Mask[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
	};
void Add(char* input, char*	
<u>-</u>	
output, int N, int	
Sb1Mask1[3][3], int	8位BMP图像。
Sb1Mask2[3][3], int	参考模板:
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
          PowerConvertion1(char*
                                 功率变换, input 是输入文件名,
void
input, char*
                   output, double
                                 output 是输出文件名。如: c = 1.2,
                                 g = 0.5, N=1。支持8位BMP图像。
c, double
              g, int
                          N, int
Sb1Mask1[3][3], int
                                 int Lap1Mask[3][3] = {
Sb1Mask2[3][3], int
                                                0, 1, 0,
Lap1Mask[3][3])
                                                 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
                                 Canny 算子。支持 BMP 图像。
void CannyEdge(char* input, char*
output)
void
               EdgeEnhance (char*
                                 边缘增强。支持 BMP 图像。
input, char* output)
void
                                 图像加密,支持8位、24位和32位
           ImageEncryption(char*
inFileName, char* outFileName, char
                                 BMP 图像。inFileName 是原图图像文
                                 件名, outFileName 是解密图像文件
key)
                                 名, key 是密钥, 如 key=255。
                                 图像解密, inFileName 是加密图像文
           ImageDecryption(char*
void
inFileName, char* outFileName, char
                                 件名, outFileName 是解密图像文件
                                 名, key 是密钥,如 key=255。支持8
key)
                                 位、24 位和 32 位 BMP 图像。
                                 图像加解密, Key 是密钥, a=1 时执行
void
      EncryptionDecryption(char*
input, char* output, int Key, int a)
                                 加密, a=0 时执行解密。支持 24 位 BMP
```

	图像。
void Encryption(char* input, char*	图像加密, input 是输入文件名,
output, int Key)	output 是输出文件名。Key 是密钥。
	支持 24 位 BMP 图像。
void Decryption(char* input, char*	图像解密, input 是输入文件名,
output, int Key)	output 是输出文件名。Key 是密钥。
	支持 24 位 BMP 图像。
void Nesting(char* Biginput, char*	图像嵌套, Biginput 是输入的大图,
Smallinput, char* output)	Smallinput 是输入的小图。支持 24 位
	BMP 图像。
void Blend(char* input1, char*	图像融合之混合化, input1 和 input2
input2, char* output)	是输入的两个要融合的图像,output
	是输出文件名。支持 24 位 BMP 图像。
void Checker(char* input1, char*	图像融合之棋盘化,input1和input2
input2, char* output)	是输入的两个要融合的图像,output
	是输出文件名。支持 24 位 BMP 图像。
void Blend1(char* input1, char*	图像融合之混合化, input1 和 input2
input2, char* output)	是输入的两个要融合的图像,output
	是输出文件名。支持 24 位 BMP 图像。
void Checker1(char* input1,char*	图像融合之棋盘化,input1和input2
input2, char* output)	是输入的两个要融合的图像,output
	是输出文件名。支持 24 位 BMP 图像。