使用说明书

目录

PPM、PGM 和 PBM 图像处理	
/UV 图像处理	
AW 图像处理	
MP 图像处理	
其它处理	

PPM、PGM 和 PBM 图像处理

ŕ	
void OTSUBinarization(char*	OTSU 二值化。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 PPMtoBMP(char* input, char*	PPM 图像转为 BMP 图像。bpp 是 BMP 图
output, int bpp)	像的色深。
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 P3PPMBlur(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** Input, char* output)	保存 PBM 图像。Input 是输入的图像数据, output 是输出文件名。支持 P4 格式的 PBM 图像。
void PGMHistogramEqualization(char*	直方图均衡化, input 是输入文件名, 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:
	<pre>typedef struct {</pre>
	unsigned char red, green, blue; //像素的颜色由 RGB(红/绿/
	蓝)表示
	<pre>} PPMPixel;</pre>
	<pre>typedef struct {</pre>
	unsigned int width, height; // 图像的宽度和高度(以像素 为单位)
	万年位) PPMPixel *data;
	// 构成图像的像素
	// 构成图像的像系 } 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
input, onar · output/	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,

• , 1 , , ,)	, 目於山文供名 士扶 DC 按子的
input, char* output)	output 是输出文件名。支持 P6 格式的 PPM 图像。
void AdjustSaturation(char*	7 7 7 7 7
	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,
	如 a=125。支持 P6 格式的 PPM 图像。
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
linput, chark output, double a)	sigma 因子,如 a=5。支持 P6 格式的 PPM
	图像。
woid Magaz Crow Filt are (ab arek	1 1 1 7
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,
.1. D	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。
	支持 P6 格式的 PPM 图像。
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
output, int threshold)	入和输出都是 PGM 文件, threshold 是
	阈值,如 threshold=125。
void Brightening(char*	彩色图像增亮,输入和输出都是 P6 格
l vora prightening (chalv	ルロ国欧伯元, 個八甲間田即足 IU 僧

input, char* output, int a)	式的 PPM 图像, a 是增亮系数, 如 a=80。
void GrayBrightening(char*	灰度图像增亮,输入和输出都是 PGM 图
input, char* output, int a)	像, a 是增亮系数, 如 a=80。
void PPMFilter(char* input, char*	彩色图像滤波,输入和输出都是 P6 格
output)	式的 PPM 文件。
void PGMGrayFilter(char*	灰度图像滤波,输入和输出都是 PGM 图
input, char* output)	像。
void PPMtoBMP(char* input, char*	PPM 图像转 BMP 图像, input 是输入文
output)	件名, output 是输出文件名。支持 P6
	格式的 PPM 图像。
void PGMOtsuThreshold(string	大津阈值法, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P5 格式的
	PGM 图像。
void	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。 低通滤波图换换复 0-10 b-6
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 器 的 大 小 , var 是 噪 声 方 差 ,
size_filter, float var, int	品的人小,var 定噪户万左,nb_iterations 是 Landweber 的迭代
nb_iterations, int nbLevels, float a, int b)	mb_iterations 定 Landweber 的这代 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,
voia Repairi (chair input, chair	国际形义,4 0.0,11 足力肝的/公扒数,

output, int M, float a)		如 M=3。
void Repair2(char* inp	ut, char*	图像修复,a=0.0,M是分解的层次数,
output, int M, float a)		如 M=3。
void MakeNois	sel(char*	制造噪声,size_filter 是低通滤波器
input, char* ou	tput, int	的宽度。
size_filter)		
void MakeNois	se2(char*	制造噪声,nb_iterations是Landweber
input, char* ou	tput, int	的迭代数,pas=1。
nb_iterations, int pas)		
void MakeNoise3(char* ou	tput, int	制造噪声, height 是输出图像的高,
height, int width, float v	ar)	width 是输出图像的宽, var 是噪声方
		差。
void MakeNois	se4(char*	制造噪声,nb_iterations是Landweber
input, char* ou	tput, int	的迭代数,pas=1。
nb_iterations, int pas)		
void ImageReconstructi	on(char*	图像重建,支持 PGM 文件。参考:
input, char* ou	tput, int	maxDepth=80, threshold=50, tx=0,
maxDepth, int thres	shold, int	ty=0 _°
tx, int ty)		

YUV 图像处理

void YUVsuperposition(char* input1, char* input2, char* output, int width, int height, unsigned char Y_BLACK, unsigned char U_BLACK, unsigned char V_BLACK)	YUV420 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。
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 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。
void YUVsuperposition(char* input1, char* input2, char* output, int width, int height, unsigned char Y_BLACK, unsigned char U_BLACK, unsigned char V_BLACK)	yuv444p 直接叠加到 yuv420p上,不做转换, Y_BLACK 、 U_BLACK 和 V_BLACK 用于将原图中的黑 色变成透明,参考: Y_BLACK=16, U_BLACK=128, V_BLACK=128。
void YUV444toYUV420(char* input, char* output, int height, int width)	YUV444 转 YUV420, height 是输入的YUV444文件的高, width 是输入的 YUV444 文 件的宽。

void YUV444toYUV420(char* input, char* output, int height, int width, int frames)	YUV444 转 YUV420, height 和width是输入文件的高和 宽,frames 是要输入文件中 操作的帧序号。
void YUVsuperposition(char* input1, char* input2, char* output, int width, int height, unsigned char Y_BLACK, unsigned char U_BLACK, unsigned char V_BLACK)	YUV444 转到 YUV420 上的叠加, Y_BLACK、U_BLACK 和V_BLACK 用于将原图中的黑色变成透明,参考:Y_BLACK=16,U_BLACK=128,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

input, int height, int width) void RAWWrite (unsigned char** input, char* output, int height, int width) void MBVQ(char* input, char* output, int width, int height) width void RAWtoPPM_red(char* input, char* output, char* output, int width, int height 是输出图像的宽和高。 RAW 转为 PPM 后提取红色通道,参考 width=4096 , height=3072 , algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举:
input, char* output, int height, int width) void MBVQ(char* input, char* output, int width, int height) void RAWtoPPM_red(char* input, char* output 是输出图像的宽和高。 void RAWtoPPM_red(char* input, char* output, int width, int height, DebayerAlgorithm algo) RAW 转为 PPM 后提取红色通道,参考 width=4096 , height=3072 , algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。
width) void MBVQ(char* input, char* output, int width, int height) width) NBVQ 效果, input 是输入文件名, output 是输出文件名。width 和height 是输出图像的宽和高。 Noid RAWtoPPM_red(char* input, char* output, int width, int height, DebayerAlgorithm algo) RAW 转为 PPM 后提取红色通道,参考 width=4096 ,height=3072 ,algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举:
void MBVQ(char* input, char* 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* input, char* output, int width, int height, DebayerAlgorithm algo) RAW12 格式。
output, int width, int height) void RAWtoPPM_red(char* input, char* output, int width, int height, DebayerAlgorithm algo) RAW 转为 PPM 后提取红色通道,参考 width 和 width algo input, char* output, int width, int height, DebayerAlgorithm algo input, E 和 以下格举:
height 是输出图像的宽和高。 void RAWtoPPM_red(char* input, char* output, int width, int height, DebayerAlgorithm algo) algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举:
void RAWtoPPM_red(char* RAW 转为 PPM 后提取红色通道,参考 input, char* output, int width, int height, DebayerAlgorithm algo) algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举:
input, char* output, int width, int height, DebayerAlgorithm algo) width=4096 , height=3072 , algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举:
height, DebayerAlgorithm algo) algo=NEARESTNEIGHBOUR 或 LINEAR。 支持 RAW12 格式。 需引入以下枚举:
支持 RAW12 格式。 需引入以下枚举:
需引入以下枚举:
enum DebayerAlgorithm {
NEARESTNEIGHBOUR,
LINEAR
};
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。
需引入以下枚举:
enum DebayerAlgorithm {
NEARESTNEIGHBOUR,
LINEAR
};
void RAWtoPPM_green2(char* RAW 转为 PPM 后提取绿色 2 通道,参
input, char* output, int width, int 考: width=4096, height=3072,
height, DebayerAlgorithm algo) algo=NEARESTNEIGHBOUR 或LINEAR。
支持 RAW12 格式。
需引入以下枚举:

	NEARESTNEIGHBOUR,
	LINEAR
The provided in); DAW ## \ DD\ C H TO ## A X \ A #
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:
	而引入以下仪字: enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
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
	了发亦格 : 目於)的 DAW 图像
<pre>void RawPowerTransformation(char* input, char* output, int width, int</pre>	幂次变换, input 是输入的 RAW 图像 文件名, output 是输出的 RAW 图像文
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
	滤波相关参数,如 M=1; mask 是滤波
	器模板。支持 RAW 图像。
	参考模板: float
	{{0.1111, 0.1111, 0.1111},
	((0.1111, 0.1111, 0.1111),
	{0. 1111, 0. 1111, 0. 1111},
	{0.1111, 0.1111, 0.1111}};
void RawImageInversion(char*	图像反相, input 是输入的 RAW 图像
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
i.d	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

	ENTER A MARKET BANKER AND ENTER A STATE OF THE PARKET
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
, , , , , , , , , , , , , , , , , , , ,	参数,如 M=1。支持 RAW 图像。
	参考模板:
	int
	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
disigned fong height)	高。
woid DawToPmp (sharet input sharet	RAW 图像转为 BMP 图像, input 是输
void RawToBmp(char* input, char*	
output, int imageWidth, int	入文件名,output 是输出文件名。支持空和京和签的图像
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)	
void MagentaGray(char* input, char*	品红灰度图像。
output, int width, int height)	** 力 +
void YellowGray(char* input, char*	黄色灰度图像。
output, int width, int height)	11 AV AVI
void Transfer(char* input, char*	传递函数。
output, int width, int height)	
void Homography(char* input1, char*	单应。
input2, char* input3, char*	
output, int width, int height, int	
newwidth, int newheight)	
<pre>void MovieEffect(char* input, char*</pre>	电影效果。
output, int width, int height)	
void 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。

height)	
void DitherMatrixMethod(char*	
input, char* output, int width, int	
height, int N)	一 一
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 TernaryGrayLevell(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	
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。
<pre>void Dewarped2(char* input, char*</pre>	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。
coeffx[12], double coeffy[12])	脱蜡规范:
coefficial, double coeffy[12]/	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};
usid Tautura Campantation 1 (sharek	纹理分割,默认 K=6,N=100。
void TextureSegmentation1(char*	纹连分割,
input, char* output, int width, int	
height, int K, int N)	(Valent Valent IIII) No. 1 and 1 and 1
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。
input, char* output, int width, int	

height)	
void ErrorDiffusion3(char*	误差扩散。
input, char* output, int width, int	
height)	
void Thin(char* input, char*	图像细化。
output, int width, int height)	
<pre>void OilPainting(char* input, char*</pre>	油画效果,默认 N=2。
output, int width, int height, int N)	
void 0ilPainting1(char*	油画效果,默认 N=2。
input, char* output, int width, int	
height, int N)	
void AverageFiltering(char*	3*3 平均值滤波。
inputfile, char* outputfile, int	
width, int height)	
void GeometricMeanFiltering(char*	3*3 几何均值滤波。
inputfile, char* outputfile, int	
width, int height)	
void MedianFiltering(char*	中值滤波。
inputfile, char* outputfile, int	
width, int height)	
void FFT(char* input, char*	FFT 函数。
output, int width, int height)	
void	低通或高通滤波。LOW_PASS=1 为低通
LowPassOrHighPassFiltering(char*	滤波,否则为高通滤波,DEGREE 为滤
input, char* output, int width, int	波程度,如 DEGREE=0。
height, int LOW_PASS, int DEGREE)	
void IFFT(char* input, char*	IFFT 函数。LOW_PASS=1 为低通滤波,
output, int width, int height, int	否则为高通滤波, DEGREE 为滤波程
LOW_PASS, int DEGREE)	度,如 DEGREE=0。
void BMPtoRAW(char*	BMP 图像转 RAW 图像。支持 24 为 BMP
inputfile, char* outputfile)	图像。
void BMPtoRAW1(char* input, char*	BMP 图像转 RAW 图像。支持 24 为 BMP
output)	图像。

BMP 图像处理

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

	图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP
	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)	K. W. D. W. H. W. W. H. W. H. W.
void GenerateImage(char*	上
output, BMPMat**	像的位数,如 type=24。
color, unsigned short type)	参考用例:
coror, unsigned short type)	多名元内: 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*	求图像的平均亮度,支持 8 位和 16 位 BMP。
input)	input 是输入文件名。
int IsBitMap(FILE *fp)	判断是否是位图。
THE ISDICMAP (TILE *IP)	7191尺日足世国。
int gotWidth (FILE vfn)	基
int getWidth(FILE *fp) int getHeight(FILE *fp)	获得图片的宽度。 获得图片的高度。

	45 17 Fr & 17 + 11 1) W
unsigned short getBit(FILE	获得每个像素的位数。
*fp)	-+- / L W. LT .L. +- LI, LY .TT
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 BlobAnalysis1(char*	Blob 分析, c1 和 c2 是与颜色相关的参数,
input, char* output, int	参考: c1=128, c2=127。支持 BMP 图像。
c1, int c2)	
void DCMtoBMP(string	DCM 图像转 BMP 图像。input 是输入文件名,
input, char* output)	output 是输出文件名。
void Ins1977 (char*	Ins1977 滤镜, input 是输入文件名, output
input, char* output, int	是输出文件名。参考: ratio=100。
ratio)	
void LOMO(char* input, char*	LOMO 滤镜,DarkAngleInput 是暗角模板图
DarkAngleInput, char*	像名,参考: ratio=100。
output, int ratio)	
void PNGGray(char*	图像灰度化, input 是输入文件名, output
input, char* output)	是输出文件名。
void PNGSpotlight(char*	聚光灯效果, input 是输入文件名, output
input, char* output, int	是输出文件名。焦点坐标
centerX, int centerY, double	(centerX,centerY) , 如 :
a, double b, double c, double	centerX=400, centerY=180; a, b, c, d, e
d, double e)	是相关参数,默认 a=100,b=100,c=160,
	d=80, e=0.5.
void PNGIllinify(char*	幻化效果, input 是输入文件名, output 是
input, char* output)	输出文件名。
void PNGWaterMark(char*	图像加水印, input1 和 input2 的尺寸必须
input1, char* input2, char*	相同。
output)	
void Short(char* input, char*	矮化特效。a=1, b=128, c=2, d=0.5,
output, int a, int b, int	depth=24。支持 24 位 BMP 图像。
c, double d, int depth)	
void Rise(char* input, char*	增高特效。a=1, b=128, c=0.5, d=2,
output, int a, int b, double	depth=24。支持 24 位 BMP 图像。
c, int d, int depth)	
void Short1(char*	矮小化特效。a=1, b=128, c=0.5, d=0.5,
input, char* output, int a, int	depth=24。支持 24 位 BMP 图像。

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

void Conversion8 (double** input, unsigned char** output); void Conversion24 (BMPMat** input, BMPMatdouble** output); void Conversion24 (BMPMatdouble** input, BMPMat** output); void CrossDenoising (BMPMat** input, BMPMat** output); +字法剔除孤立像素点。 需引入以下结构体和声明: typedef struct { unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量 unsigned char G; //24 位和 32 位 BMP 图像的绿色通道分量 unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的 Alpha 通道 } BMPMat; typedef struct { double B; double G;
void Conversion24(BMPMat** input, BMPMatdouble** output); void Conversion24(BMPMatdouble** input, BMPMat** output); void CrossDenoising(BMPMat** input, BMPMat** output); void CrossDenoising(BMPMat** input, BMPMat** output); +字法剔除孤立像素点。 需引入以下结构体和声明: typedef struct { unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量 unsigned char G; //24 位和 32 位 BMP 图像的绿色通道分量 unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的 Alpha 通道 } BMPMat; typedef struct { double B;
input, BMPMatdouble** output); void CrossDenoising(BMPMat** input, BMPMat** output); void CrossDenoising(BMPMat** input, BMPMat** output); +字法剔除孤立像素点。 需引入以下结构体和声明: typedef struct { unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量 unsigned char G; //24 位和 32 位 BMP 图像的绿色通道分量 unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 35 位 BMP
void CrossDenoising(BMPMat** input, BMPMat** output); void CrossDenoising(BMPMat** + 字法剔除孤立像素点。 input, BMPMat** output, double a)
void CrossDenoising(BMPMat** input,BMPMat** output,double a) typedef struct { unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量 unsigned char G; //24 位和 32 位 BMP 图像的绿色通道分量 unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图像的红色通道分量 typedef struct { double B;
roid CrossDenoising(BMPMat** input,BMPMat** output, double a) ### ### ### #### ##################
roid CrossDenoising(BMPMat** input,BMPMat** output, double a) ### ### ### #### ##################
input, BMPMat** output, double a)
typedef struct { unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量 unsigned char G; //24 位和 32 位 BMP 图像的绿色通道分量 unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量 unsigned char A; //仅限 32 位 BMP 图 像的 Alpha 通道 } BMPMat; typedef struct { double B;
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;
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;
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;
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;
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;
unsigned char A; //仅限 32 位 BMP 图像的 Alpha 通道}BMPMat; typedef struct { double B;
unsigned char A; //仅限 32 位 BMP 图像的 Alpha 通道}BMPMat; typedef struct { double B;
像的 Alpha 通道 } BMPMat; typedef struct { double B;
<pre>}BMPMat; typedef struct { double B;</pre>
<pre>typedef struct { double B;</pre>
double B;
double B;
· ·
I dollble (1;
double R;
double A;
}BMPMatdouble;
void Conversion8(unsigned char**
<pre>input, double** output);</pre>
void Conversion8(double**
input, unsigned char** output);
void Conversion24(BMPMat**
input, BMPMatdouble** output);
void Conversion24(BMPMatdouble**
input, BMPMat** output);
CrossConnectionDenoising (uns
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;
                              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);
                              交叉法剔除孤立像素点。
void
CrossConnectionDenoising (BMP
                              需引入以下结构体和声明:
              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;
                             void
                                     Conversion8 (unsigned
                                                            char**
                              input, double** output);
                             void
                                              Conversion8 (double**
                              input, unsigned char** output);
                                             Conversion24 (BMPMat**
                              void
                              input, BMPMatdouble** output);
```

	void Conversion24(BMPMatdouble**
	input,BMPMat** output);
void	矩阵法剔除孤立像素点。
MatrixDenoising(unsigned	需引入以下结构体和声明:
char** input, unsigned char**	typedef struct {
output, double a)	unsigned char B; //24 位和 32 位 BMP 图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP
	图像的绿色通道分量
	unsigned char R; //24 位和 32 位 BMP 图像的红色通道分量
	unsigned char A; //仅限 32 位 BMP 图
	像的 Alpha 通道
	BMPMat;
) bill like t,
	typedef struct {
	double B;
	double G;
	double G;
	·
	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;
	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 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
<pre>dx[], int dy[], int a, double</pre>	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
	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
	称。
void Dark(char* input, char*	暗调滤镜,参考: ratio=100。
output, int ratio)	
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
J ,	/// P =

	生成 TGA 图像,参考: degree=10。
void PinchFilter(char*	挤压形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int a)	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
imput, char voutput, int a)	a=3 时生成 TGA 图像。
Dinah Filtar (abarak	
void PinchFilter(char*	挤压形变特效滤镜,a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像,a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
	坐标, cenY 是形变中心点 Y 坐标。
void SpherizeFilter(char*	球面形变特效滤镜,a=0 时生成 BMP 图像,
input, char* output, int a)	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
.1. 0.1	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 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(IIOI IIIO
string input, char* output)	
CImg unsigned int	HSI 空间。
HistEqualColorImageHSISpace(1101 11110
string input)	
void ColorTransfer1(char*	 颜色转移。
sourceImage, string	
targetImage, char* output)	
CImg <unsigned int=""></unsigned>	颜色转移。
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
	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*	
input, char* output, int	
ratio)	
void	生成图像的灰度图,支持 8 位 BMP 图像。
GrayImageConversion8(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。
void Gray(char* input,char*	灰度图转换,支持24位BMP图像。input是
output)	输入文件名, output 是输出文件名。
void	彩色图转灰度图, input 是要处理的彩色图
GrayImageConversion(char*	像, output 是处理后生成的灰度图名称。支
input, char* output)	持 24 位 BMP 图像。
void	二值图像垂直镜像, input 是输入图像的像
BinaryImageVerticalMirror(un	素数据, output 是输出图像的像素数据, w
signed char *input,unsigned	是输入图像的宽,h 是输入图像的高。
char *output,unsigned int	

w, unsigned int h)	
void	 灰度图像垂直镜像, input 是输入图像的像
GrayImageVerticalMirror(unsi	素数据,output 是输出图像的像素数据,w
gned char *input, unsigned	是输入图像的宽,h是输入图像的高。
	定栅八图像的见,II 定栅八图像的同。
<u> </u>	
w, unsigned int h)	彩色图像垂直镜像, input 是输入图像的像
	素数据,output 是输出图像的像素数据,w
ColorImageVerticalMirror(uns igned char *input, unsigned	是输入图像的宽,h 是输入图像的高。
char *output, unsigned int	定制八国家的见,II 定制八国家的问。
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)	六年异宏,Input 是襉八叉杆石,output 是 输出文件名。BeforeThreshold 是初始阈
output, Int before in eshord)	- 捕出文件名。Before Threshold=10。支持 8 位 BMP
	恒,如 before fill eshord—10。文持 6 位 bmr 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)	前山文[十石。文]
void HightBrightness(char*	调高亮度, input 是输入文件名, output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100, b=0。
void	迭代阈值选择, input 是输入文件名,
IterativeThresholdSelection(output 是输出文件名。支持 8 位 BMP 图像。
char* input, char* output)	。
void DitheringMethod(char*	抖动法, input 是输入文件名, output 是输
input, char* output)	出文件名。支持8位BMP图像。
void LogTransformation(char*	对数变换, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。constant
constant)	是相关参数,如 constant=15。
void	对数变换, input 是输入文件名, output 是
LogarithmicTransformation(ch	输出文件名。支持 BMP 图像。
ar* input, char* output)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization(char*	output 是输出文件名。支持 BMP 图像。
input, char* output)	
void Binarization(char*	二值化, input 是输入文件名, output 是输
input, char* output, int	出文件名。支持 24 位 BMP 图像。threshold
threshold)	是阈值,如: threshold=128。
void Expansion(char*	二值图像膨胀,参考:
input, char* output, unsigned	mask[9]={0,255,0,255,255,255,0,255,0}
char mask[9], int c)	, c=128。
void Corrosion(char*	二值图像腐蚀,参考:
input, char* output, unsigned	mask[9]={0,255,0,255,255,255,0,255,0}
char mask[9], int c)	, c=128。

void OpenOperation(char*	二值图像开运算,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128°
<pre>void ClosedOperation(char*</pre>	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°
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
unsigned char**	TempBuf[4][4]。 线性存储的灰阶图像像素转化为二维。

create2DImg(unsigned char*	
input, int w, int h)	
unsigned char	图像指定区域取最大值(判断是否超出边
getMaxPixelWhole(unsigned	界)。
char **input, int x, int y, int	
w, int h, int *Kernal, int	
kernalW, int halfKernalW)	
unsigned char	图像指定区域取最大值(不判断是否超出边
getMaxPixelCenter(unsigned	界)。
char **input, int x, int y, int	
*Kernal, int kernalW, int	
halfKernalW)	
unsigned char**	图像膨胀。
imgDilate(unsigned char	
*input, int w, int h, int	
*Kernal, int kernalW, int	
halfKernalW)	
unsigned char	图像指定区域取最小值(判断是否超出边
getMinPixelWhole(unsigned	界)。
char **input, int x, int y, int	
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	7170
*Kernal, int kernalW, int	
halfKernalW)	
unsigned char**	图像腐蚀。
imgErode (unsigned char	E BON A
*input, int w, int h, int	
*Kernal, int kernalW, int	
halfKernalW)	
void Corrosion (unsigned char	二值腐蚀。
*input, unsigned char	— IE-/M KAIV
*output, int rows, int	
cols, int mat[5][5])	
void Expansion (unsigned char	二值膨胀。
*input, unsigned char	→ IFT NAV UNC 0
*output, int rows, int	
cols, int mat[5][5])	
void	高斯滤波,支持 PNG 文件。
GaussianBlurFilter(char*	PH/YIT/WUTZ, ZNINOZH
input, char* output)	
	享斯滤波 input 具輪 λ 立州夕 output 見
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 是輸入的原图像,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
input, char output)	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
input, char " output)	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*	
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)	2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
void GreyPesudoColor(char*	灰度图伪彩色化, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
void	计算累加直方图并映射, input 是输入文件
CalculateCumulativeHistogram	名,outfile 是输出文件名。支持 24 位 BMP
Map(char* input, char*	图像。
outfile)	
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

	接海提 <u>炸</u> 后的灶里文件名 : 具接换亦
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)	1 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
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)	1 /4 - MILIA - HALL A Fac > 4 - HI 4/2
void Conversion8 (unsigned	unsigned char**转 float **, output 用于
char** input, float** output)	保存结果(与 input 大小相同)。
void Conversion8(float**	float **转 unsigned char**, output 用于
input, unsigned char**	保存结果(与 input 大小相同)。
output)	
void Conversion8 (unsigned	unsigned char**转double **, output用
char** input, double**	于保存结果(与 input 大小相同)。
output)	The state of the s
ou opa o/	

void Conversion8(double**	double **转 unsigned char**, output 用
input, unsigned char**	于保存结果(与 input 大小相同)。
output)	1 NVILSHW (—) INDAR VOLUTER)
	unsigned char**转 char **, output 用于
, 0	
char** input, char** output)	保存结果(与 input 大小相同)。
void Conversion8(char**	char **转 unsigned char**, output 用于
input, unsigned char**	保存结果(与 input 大小相同)。
output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatshort **, output 用于
input, BMPMatshort** output)	保存结果(与 input 大小相同)。
void	BMPMatshort **转 BMPMat **, output 用于
Conversion24(BMPMatshort**	保存结果(与 input 大小相同)。
input,BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatint **, output 用于保
input, BMPMatint** output)	存结果(与 input 大小相同)。
void	BMPMatint **转 BMPMat **, output 用于保
Conversion24(BMPMatint**	存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatfloat **, output 用于
<pre>input, BMPMatfloat** output)</pre>	保存结果(与 input 大小相同)。
void	BMPMatfloat **转 BMPMat **, output 用于
Conversion24(BMPMatfloat**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转BMPMatdouble **, output用
input,BMPMatdouble** output)	于保存结果(与 input 大小相同)。
void	BMPMatdouble **转BMPMat **, output用
Conversion24(BMPMatdouble**	于保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatchar **, output 用于
input, BMPMatchar** output)	保存结果(与 input 大小相同)。
void	BMPMatchar **转 BMPMat **, output 用于
Conversion24(BMPMatchar**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion32 (BMPMat**	BMPMat **转 BMPMatshort **, output 用于
input, BMPMatshort** output)	保存结果(与 input 大小相同)。
void	BMPMatshort **转 BMPMat **, output 用于
Conversion32(BMPMatshort**	保存结果(与 input 大小相同)。
input, BMPMat** output)	NEW YORK A TUBER OF A URLAND
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)	 1年2日本(コ IIIhat ソクス目的)。
	DMDMot *** DMDMotfloot *** output 田工
void Conversion32(BMPMat**	BMPMat **转 BMPMatfloat **, output 用于

input, BMPMatfloat** output) void Conversion32 (BMPMatfloat** input, BMPMat** output) void Conversion32 (BMPMat**) input, BMPMat** output) void Conversion32 (BMPMat**) input, BMPMatdouble** output) void Conversion32 (BMPMatdouble**; input, BMPMat** output) void Conversion32 (BMPMatdouble**; input, BMPMat** output) void Conversion32 (BMPMat**) input, BMPMat** output) void Conversion32 (BMPMat**) input, BMPMat** output) void MeanFiltering (char** input, char* output) void MeanFiltering (char* input, char* output, int BeforeThreshold) BeforeThreshold Bef		
Conversion32 (BMPMatt*) input, BMPMat** output) void Conversion32 (BMPMatwoutput) void Conversion32 (BMPMatdouble** output) void Conversion32 (BMPMatdouble** input, BMPMat** output) void Conversion32 (BMPMat** output) void Conversion32 (BMPMat** output) void Conversion32 (BMPMat** output) void Conversion32 (BMPMat** output) void Conversion32 (BMPMatchar** output) void MeanFiltering (char* input, char* output) void OpenOperation (char* input, char* output, intratio) void OpenOperation (char* input, char* output, intratio) void LapulasFiltering (char* vritePath, float CoefArray[9], float coef) void LapulasFiltering (char* vr	voia	DMDMo+floot ** DMDMo+ ** OUTPUT 用工
input, BMPMat** output) void Conversion32 (BMPMat*** bMPMat double ***, output 用于保存结果(与 input 大小相同)。 void Conversion32 (BMPMatdouble*** input, bMPMat** output) void Conversion32 (BMPMat*** bMPMat** output) void Conversion32 (BMPMat** input, bMPMat** output) void Conversion32 (BMPMatehar** input, bMPMat** output) void MeanFiltering (char* input, char* output, int ratio) void OpenOperation (char* input, char* output) void LapulasFiltering (char* input, char* output, int ratio) void LapulasFiltering (char* input, cha	Convergion 22 (DMDMotflooty	
Void Conversion32 (BMPMat** input, BMPMatdouble** output)		体行组术(与 Imput 人小相同)。
input, BMPMatdouble** output) void Conversion32 (BMPMatdouble** input, BMPMat** output) void Conversion32 (BMPMat** input, BMPMatchar** output) void Conversion32 (BMPMat** input, BMPMatchar** output) void Conversion32 (BMPMatchar** input, BMPMatchar** output) void MeanFiltering (char* input, char* output) void KapoorAlgorithm (char* input, char* output, int BeforeThreshold) void OpenOperation (char* input, char* output) void Diffusion (char* input, char* output) void LapulasFiltering (char* readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) void ImageFiltering (char* input, char* output, int ratio) void ImageFiltering (char* input, char* output, char* output, char* output, char* output, char* output, int ratio) void LapulasFiltering (char* input, char* output, char* output, int ratio) void LapulasFiltering (char* input, char* output, char* out		DIDIV 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
BMPMatdouble **转 BMPMat ***, output 用于保存结果(与 input 大小相同)。		1
Terrest (与 input 大小相同)。 Terrest (与 input 大小相同)。 Terrest (与 input 大小相同)。 Terrest (与 input 大小相同)。 Terrest (P input 大相同)。 Terrest (P input 大小相同)。 Terrest (P input 大相同)。 T		
input, BMPMat** output) void Conversion32 (BMPMat** binput, BMPMatchar** output) void Conversion32 (BMPMatchar** output) void Conversion32 (BMPMatchar** output) void MeanFiltering (char* input, char* output) void MeanFiltering1 (char* input, char* output) void MeanFiltering1 (char* input, char* output) void MeanFiltering1 (char* input, char* output) void KapoorAlgorithm (char* input, char* output, int input, char* output, int patt, char* output) void OpenOperation (char* input, char* output) void Diffusion (char* input, char* output, int ratio) void LapulasFiltering (char* readPath, char* writePath, float CoefArray[9], float coef) void ImageFiltering (char* input, char* output, int ratio) void LapulasFiltering (char* readPath, char* writePath, float CoefArray[9] = {1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f}; 定义模板前乘的系数(拉普拉斯):float CoefArray[9]={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* input, char* output, float Void ImageFiltering (char* input, char* output, char* output, float Void ImageFiltering (char* input, char* output, char* output, char* output, char* output, char* output, c		_
woid Conversion32 (BMPMat** input, BMPMatchar** output) void void Conversion32 (BMPMatchar** output) void MeanFiltering(char* input, char* output) void KapoorAlgorithm(char* input, char* output, int peforeThreshold) void OpenOperation(char* input, char* output) void Diffusion(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) void ImageFiltering(char* input, char* output, float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 of, 2.0f, 2.0f, 1.0f, 2.0f, 4.0 of, 2.0f, 2.0		丁保仔结果(与 1nput 大小相同)。
rinput, BMPMatchar** output) void void Conversion32 (BMPMatchar** input, BMPMat** output) void MeanFiltering (char* input, char* output) void MeanFltering1 (char* input, char* output) void KapoorAlgorithm(char* input, char* output, int BeforeThreshold) void OpenOperation (char* input, char* output) void Diffusion (char* input, char* output) void LapulasFiltering (char* readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void ImageFiltering (char* input, char* output, float Coef=(float) (1.0/16.0); Void Im		
BMPMatchar **转 BMPMat **, output 用于保存结果(与 input 大小相同)。		1
Conversion32 (BMPMatchar** input, BMPMat** output) void MeanFiltering (char* input, char* output) void MeanFltering1 (char* 物出文件名。支持8位BMP 图像。 void MeanFltering1 (char* 物出文件名。支持8位BMP 图像。 void KapoorAlgorithm(char* input, char* output, int BeforeThreshold) void OpenOperation (char* input, char* output) void Diffusion (char* input, char* output, int ratio) void LapulasFiltering (char* writePath, float CoefArray[9], float coef) void ImageFiltering (char* input, char* output, float Void ImageFiltering (char* input, char* output, char*		
input, BMPMat** output) void MeanFiltering(char* input, char* output) void MeanFltering1(char* 物出文件名。支持8位BMP 图像。 void MeanFltering1(char* 物出文件名。支持8位BMP 图像。 void KapoorAlgorithm(char* input, char* output, int BeforeThreshold) void OpenOperation(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* readPath, char* writePath, float CoefArray[9], float coef) void ImageFiltering(char* input, char* output, int ratio void ImageFiltering(char* input, char* output, int ratio) void ImageFiltering(char* input, char* output, float void ImageFiltering(char* input, char* ou	void	
yoid MeanFiltering(char* 均值滤波, input 是输入文件名, output 是输出文件名。支持 8 位 BMP 图像。 yoid MeanFiltering1(char* 均值滤波, input 是输入文件名, output 是输出文件名。支持 8 位 BMP 图像。 yoid KapoorAlgorithm(char* input, char* output, int BeforeThreshold) woid OpenOperation(char* input, char* output) void OpenOperation(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* yreadPath, char* writePath, float CoefArray[9], float coef) woid ImageFiltering(char* input, char* output, float Coef=(float)(1.0/16.0); void ImageFiltering(char* input, char* output, float woid ImageFiltering(char* input, char* output, float wide input, ehar output, ehar output, ehar output, ehar output, float wide input, ehar output, ehar output, ehar output, ehar output, float wide input, ehar output,	Conversion32(BMPMatchar**	保存结果(与 input 大小相同)。
input, char* output) void MeanFltering1 (char* juput, char* output) void KapoorAlgorithm(char* input, char* output, int input, char* output, int BeforeThreshold) void OpenOperation(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* writePath, float CoefArray[9], float coef) void ImageFiltering(char* input, char* output, float Void ImageFiltering(char* input, char* output, float Coef (float) (1.0/16.0); void ImageFiltering(char* input, char* output, float Void ImageFiltering(char* input, char* output, char* output, char* output, char* output, char* input, char* output,	input,BMPMat** output)	
voidMeanFlteringl(char* input, char* output)均值滤波, input 是输入文件名, output 是输出文件名。支持 8 位和 24 位 BMP 图像。voidKapoorAlgorithm(char* input, char* output, int input, char* output, int input, char* output)是输出文件名。BeforeThreshold是初始阈值,如 BeforeThreshold=150。支持 8 位 BMP 图像。voidOpenOperation(char* input, char* output)开运算, input 是输入文件名, output 是输 input, char* output, intratio)voidDiffusion(char* input, char* output, intratio)扩散滤镜,参考: ratio=90。voidLapulasFiltering(char* readPath, char* writePath, float拉普拉斯滤波, readPath 是原图像, writePath 是处理后的图像文件名。支持 8 位 BMP 图像。CoefArray[9], float coef)各参数参考值: 定义*3 的模板(拉普拉斯): floatCoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f, 2.0f, 4.0 foat coef=(float)(1.0/16.0);voidImageFiltering(char* input, char* output, float图像滤波, input 是输入文件名, output 是输出文件名。kernel 是模糊内核。支持 24	void MeanFiltering(char*	均值滤波, input 是输入文件名, output 是
input, char* output) wid KapoorAlgorithm(char*	input, char* output)	输出文件名。支持 8 位 BMP 图像。
roid KapoorAlgorithm(char*	void MeanFltering1(char*	均值滤波, input 是输入文件名, output 是
input, char* output, int BeforeThreshold) BeforeThreshold) BeforeThreshold) BeforeThreshold) BeforeThreshold) BeforeThreshold = 150。支持 8 位 BMP 图像。 void OpenOperation(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) Abbeta	input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
BeforeThreshold) (值,如 BeforeThreshold=150。支持 8 位 BMP 图像。 void OpenOperation(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) CoefArray[9], float coef) Above the path of the path	void KapoorAlgorithm(char*	卡普尔算法, input 是输入文件名, output
woid OpenOperation(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) woid ImageFiltering(char* input, char* output, int ratio) Void LapulasFiltering(char* readPath 是原图像, writePath 是处理后的图像文件名。支持8 位 BMP 图像。 CoefArray[9], float coef) Float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 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* input, char* output, float 输出文件名。kernel 是模糊内核。支持24	input, char* output, int	是输出文件名。BeforeThreshold 是初始阈
void OpenOperation(char* input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) void ImageFiltering(char* input, char* output, int ratio) void LapulasFiltering(char* input, char* writePath 是处理后的图像文件名。支持8 位 BMP 图像。 各参数参考值:定义*3 的模板(拉普拉斯):float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f, 2.0f, 1.0f};定义模板前乘的系数(拉普拉斯):float coef=(float)(1.0/16.0); void ImageFiltering(char* input, char* output, float 物出文件名。kernel 是模糊内核。支持24	BeforeThreshold)	值,如 BeforeThreshold=150。支持 8 位 BMP
input, char* output) void Diffusion(char* input, char* output, int ratio) void LapulasFiltering(char* 控音力 是原图像,writePath, char* writePath 是处理后的图像文件名。支持8 位 BMP 图像。 CoefArray[9], float coef) CoefArray[9], float coef) Above the provided House of the provided House output, float output, char* output, float output, fl		图像。
void Diffusion(char* が散滤镜,参考: ratio=90。 input, char* output, int ratio) void LapulasFiltering(char* 拉普拉斯滤波,readPath 是原图像, readPath, char* writePath 是处理后的图像文件名。支持 8 writePath, float CoefArray[9], float coef) 各参数参考值: 定义*3 的模板(拉普拉斯): float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f, 2.0f, 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* input, char* output, float 物出文件名。kernel 是模糊内核。支持 24	void OpenOperation(char*	开运算, input 是输入文件名, output 是输
input, char* output, int ratio) void LapulasFiltering(char* 拉普拉斯滤波, readPath 是原图像, writePath, char* writePath 是处理后的图像文件名。支持 8 位 BMP 图像。 CoefArray[9], float coef) About CoefArray[9], float coef) About 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* input, char* output, float with an analysis of the part output, float with an analysis output, float with an analysis of the part output, input, char* output, input, char* output, float with an analysis output, input, char* output, input, char* output, float with an analysis output, input, char* output, input, input, char* output, float with an analysis output, input, char* output, float with an analysis output, input, char* output, float with an analysis output, input, input, char* output, float with an analysis output, input, in	input,char* output)	出文件名。支持 4 位 BMP 图像。
ratio)	void Diffusion(char*	扩散滤镜,参考: ratio=90。
void LapulasFiltering(char* readPath, char* writePath, float	input, char* output, int	
readPath, char* writePath, float CoefArray[9], float coef) CoefArray[9], float coef) A参数参考值: 定义*3 的模板(拉普拉斯): float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f, 2.0f, 1.0f}; 定义模板前乘的系数(拉普拉斯): float coef=(float)(1.0/16.0); void ImageFiltering(char* input, char* output, float WritePath 是处理后的图像文件名。支持8 位 BMP 图像。 A参数参考值: 定义*3 的模板(拉普拉斯): float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f, 2.0f, 1.0f}; 定义模板前乘的系数(拉普拉斯): float coef=(float)(1.0/16.0);	ratio)	
writePath, float CoefArray[9], float coef) CoefArray[9], float coef) 名参数参考值: 定义*3 的模板(拉普拉斯): float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0} f, 2.0f, 1.0f, 2.0f, 1.0f}; 定义模板前乘的系数(拉普拉斯): float coef=(float)(1.0/16.0); void ImageFiltering(char*	void LapulasFiltering(char*	拉普拉斯滤波, readPath 是原图像,
writePath, float CoefArray[9], float coef) CoefArray[9], float coef) 名参数参考值: 定义*3 的模板(拉普拉斯): float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f, 2.0f, 1.0f}; 定义模板前乘的系数(拉普拉斯): float coef=(float)(1.0/16.0); void ImageFiltering(char* input, char* output, float 输出文件名。kernel 是模糊内核。支持 24	readPath,char*	writePath 是处理后的图像文件名。支持 8
定义*3 的模板(拉普拉斯): float CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0 f, 2.0f, 1.0f, 2.0f, 1.0f}; 定义模板前乘的系数(拉普拉斯): float coef=(float)(1.0/16.0); void ImageFiltering(char* 图像滤波, input 是输入文件名, output 是 input, char* output, float 输出文件名。kernel 是模糊内核。支持 24	writePath,float	
float	CoefArray[9], float coef)	各参数参考值:
float		定义*3 的模板(拉普拉斯):
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*		
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*		CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0
c定义模板前乘的系数 (拉普拉斯): float coef=(float) (1.0/16.0);voidImageFiltering (char* input, char*图像滤波, input 是输入文件名, output 是 输出文件名。kernel 是模糊内核。支持 24		
float coef=(float)(1.0/16.0); void ImageFiltering(char* 图像滤波, input 是输入文件名, output 是input, char* output, float 输出文件名。kernel 是模糊内核。支持 24		
void ImageFiltering(char* 图像滤波, input 是输入文件名, output 是input, char* output, float 输出文件名。kernel 是模糊内核。支持 24		
input, char* output, float 输出文件名。kernel 是模糊内核。支持 24	void ImageFiltering(char*	图像滤波, input 是输入文件名, output 是
Kerner[o][o]/ 「仏 DMF 宮頂。	kerne1[3][3])	位 BMP 图像。
void ComicStrip(char* 连环画滤镜,参考: ratio=100。	void ComicStrip(char*	连环画滤镜,参考: ratio=100。
input, char* output, int	input, char* output, int	
ratio)		
void 亮度对比度调节,参考: brightness=-30,	void	亮度对比度调节,参考: brightness=-30,

	100
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	Contrast 1000
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)	
	泥业滤焙 innut 和 Mool
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},
	$\{2, 4, 2\},$
	$\{1, 2, 1\}\};$
:1 W 1: D:1, : / 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] = $\{\{1, 2, 1\}, \dots \}$
sharpen[3][3])	$\{2, 4, 2\},$
	{1, 2, 1}};
	short sharpen[3][3] = $\{\{-1, -1, -1\},$
	$\{-1, 8, -1\},$
	$\{-1, -1, -1\}\};$
void	梯度锐化, input 是输入文件名, output 是
GradientSharpening(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output, short	参考模板:
average[3][3], short	short average[3][3] = {{1, 2, 1},
	_
soble1[3][3], short	$\{2, 4, 2\},$
soble2[3][3])	{1, 2, 1}};
	short soble1[3][3] = $\{\{-1, -2, -1\},$

	$\{0, 0, 0\},\$ $\{1, 2, 1\}\};$ short soble2[3][3] = \{\{-1, 0, 1\},\} $\{-2, 0, 2\},\$ $\{-1, 0, 1\}\};$
void	算术平均滤波器, input 是输入文件名,
ArithmeticMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
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)	1 7 2 1111 7 2 1111 7 2 1 1 1 1 1 1 1 1
void	调和平均滤波器, input 是输入文件名,
HarmonicMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	output Cinial XII I o XII o E Zim Eliko
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)	output Emilia en a en
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
· 1	的大小。支持 24 位 BMP 图像。
void MosaicFilter(char*	马赛克滤镜,参考: ratio=50。
input, char* output, int ratio)	
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。
	中值滤波器, input 是输入文件名, output
void MidSmoothing(char*	1 / 2 / 2 / 2 / 2 / 2 / 2 / 2 / 2 / 2 /
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

	46 15 17 11 to 17 15 15 15 15 15 15 15 15 15 15 15 15 15
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	$\{0.0, 0.0, 1.0\}, \text{ vector_s}[3] = \{0.0,$
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)	SA FIRE LITTLE IN
void	彩色图像水平镜像。
ColorImageHorizontalMirror(u	
nsigned char *input, unsigned	

1	
char *output, unsigned int w, unsigned int h)	
void SketchFilter(char*	素描滤镜,参考: ratio=100。
input, char* output, int	, , , , , , , , , , , , , , , , , , ,
ratio)	
void Zoom(char* input, char*	缩放,参考: scaleX=5, scaleY=5,
output, float scaleX, float	interpolation=0或interpolation=1。
scaleY, int interpolation)	
void AddGaussNoise(char*	添加高斯噪声, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
void	添加椒盐噪声, input 是输入文件名,
AddSaltPepperNoise(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void ChannelSeparation(char*	通道分离, input 是输入文件名, Routput
input, char* Routput, char*	是红色通道图像, Goutput 是绿色通道图
Goutput, char* Boutput)	像, Boutput 是绿色通道图像。支持 24 位
	BMP 图像。
void PatternMethod(char*	图案法, input 是输入文件名, output 是输
input, char* output, unsigned	出文件名。Template 是模板数组。支持 8 位
char Template[8][8])	BMP 图像。
void	图层算法, input 是基底图层图像,
LayerAlgorithm(char*input,ch	inputMix 是混合图层图像,参考:
ar* inputMix, char*	alpha=50, blendModel=26。
output, int alpha, int	blendModel 的取值对应的模式如下:
blendModel)	1 典型
	2 溶解
	3 暗化
	4 多层
	5 颜色加深模式
	6 线性加深
	7 暗调
	8 亮化
	9 遮盖
	10 颜色减淡模式
	11 线性减淡
	12 浅色
	13 叠加
	14 柔光模式
	15 强光模式
	16 艳光模式
	17 线性光模式
	18 点光模式
	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
BMP24LosslessCompression(cha	名, output 是无损压缩后输出的文件名。支
r* input, char* output)	右, output 足光频虚组启制出的文件名。文 持 24 位 BMP 图像。
void	图像无损解压,input 是待解压的文件名,
BMP24LosslessDecompression(c	output 是输出解压后的 BMP 文件名。支持
	Output 定抽出解压后的 bMF 文件名。文持 24 位 BMP 图像。
har* input, char* output)	
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**	图像变形之水平内凹,返回处理结果。参考:
HorizontalConcavity(unsigned	RANGE=400。
char** input, int RANGE, int	
height, int width)	
unsigned char**	图像变形之水平外凸,返回处理结果。参考:
HorizontalConvexity(unsigned	RANGE=400。
char** input, int RANGE, int	
height, int width)	
unsigned char**	图像变形之梯形形变,返回处理结果。参考:
TrapezoidalDeformation(unsig	k=0.3°
ned char** input, int	
height, int width, double k)	
unsigned char**	图像变形之三角形形变,返回处理结果。参
TriangularDeformation(unsign	考: k=0.5。
ed char** input, int	
height, int width, double k)	
unsigned char**	图像变形之 S 形变,返回处理结果。参考:
SDeformation(unsigned char**	RANGE=450。
input, int height, int	
width, int RANGE)	
int LsdLineDetector(unsigned	LSD 直线检测器。
, 0	et eter

```
图像,单通道
char *src, int w, int h, float
                             [in]
                                       src:
                                                      宽
scaleX,
           float
                    scaleY,
                             [in]
                                       w:
                                                      高
boundingbox t
                             [in]
                                       h:
                      bbox,
std::vector<line float t>
                             [in]
                                     scaleX:
                                                X轴上的缩小因子
&lines)
                             [in]
                                     scaleY:
                                                Y轴上的缩小因子
                             [in]
                                     bbox:
                                                 要检测的边界框
                             [in/out] lines:
                                                  结果
                                                 0:ok; 1:error
                             return:
                             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
                             [in]
                                                      宽
                                       w:
h, float scaleX, float scaleY,
                             [in]
                                                      高
                                       h:
                                                X轴上的缩小因子
boundingbox t
                      bbox,
                             [in]
                                    scaleX:
std::vector<line_float_t>
                             [in]
                                    scaleY:
                                                Y轴上的缩小因子
&lines)
                             [in]
                                     bbox:
                                                 要检测的边界框
                                                  结果
                             [in/out] lines:
                                                 0:ok; 1:error
                             return:
                             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	传播滤波器。
PropagatedFilter1(unsigned	[in] src: 输入图像
char *src, unsigned char	
*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	传播滤波器。
PropagatedFilter2(unsigned	[in] src: 输入图像
char *src, unsigned char	
*guidance, unsigned char	
*dst, int w, int h, int c, int	
r, float sigma_s, float	
sigma r)	[in] c: 图像通道,仅 c=1 或 c=3
516ma_1/	[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	
char *src, unsigned char	l[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	[in/out] dst: 输出图像,单通道
*dst, int w, int h, int c, int	[in] w: 宽
r)	[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	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] 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: td="" 向上缩放<=""></sr<1:>
	如果正则化, _scale = 1; 如果不正则化,
	scale = 255*255
	return: 0:ok; 1:error
	eg: $r = 4$, (try sr = $r/4$ to sr=r), (try
	rp=0.1^2, 0.2^2, 0.4^2)
	try: (src, guidance, dst, w, h, 1, 4, 0.01, 4,
	255*255)
	condition: (MIN(w, h) / sr) > 1
	condition: (int) $(r / sr + 0.5f) >= 1$
int	快速导向滤波
fast guided filter1(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] 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: td="" 向上缩放<=""></sr<1:>
	-III // \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \

如果正则化, scale = 1; 如果不正则化, scale = 255*255return: 0:ok; 1:error eg: r = 4, (try sr = r/4 to sr=r), (try rp=0.1², 0.2², 0.4²) try: (src, guidance, dst, w, h, 1, 4, 0.01, 4, 255*255) condition: (MIN(w, h) / sr) > 1condition: (int) (r / sr + 0.5f) >= 1霍夫线探测器。 HoughLineDetector (unsigned [in] src: 图像,单通道 char *src, int w, int h, float 宽 [in] w: scaleX, float scaleY, float [in] 高 h: X轴上的缩小因子 CannyLowThresh, float [in] scaleX: Y轴上的缩小因子 CannyHighThresh, float [in] scaleY: HoughRho, float HoughTheta, [in] CannyLowThresh: canny 算子中迟滞 float 过程的低阈值 MinThetaLinelength, 「in Canny High Thresh: canny 算子中迟滞 float MaxThetaGap, int HoughThresh, HOUGH_LINE_TYPE_ 过程的高阈值 [in] HoughRho: 累加器的距离分辨率(以 CODE type, boundingbox t bbox. 像素为单位) std::vector<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] MaxThetaGap: 标准:对于标准和 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) 线 方 法: type: hough hough line STANDARD 或 hough line PROBABILISTIC [in] 要检测的边界框 bbox: [in/out] lines: 结果 return 0:ok: 1:error 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.
                             PI/180,
                                          0.
                                                             100,
                             HOUGH LINE STANDARD, bbox, line)
                                  于
                                      概
                                           率
                                                的
                                                              try
                             (src, w, h, scalex, scaley, 70, 150,
                                                               1,
                             PI/180,
                                           30,
                                                    10,
                                                              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
echannel (unsigned char *src,
                             [in]
unsigned
          char
                 *guidance,
                             [in/out]
                                      dst:
                                              输出图像,单通道
unsigned char *dst, int w,
                             [in]
                                               宽
                                        w:
int h, float sigma s, float
                             [in]
                                              高
                                       h:
sigma r, float
                             [in]
              scale)
                                     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*255return: 0:ok; 1:error 快速双边滤波器单通道。 输入图像,单通道 fast_bilateral_filter_single [in] src: guidance: 引导图像,单通道 channel (unsigned char *src, [in] 输出图像,单通道 unsigned char *guidance, [in/out] dst: [in] unsigned char *dst, int w, 宽 w:高 int h, int c, float sigma s, [in] h: 图像通道,仅 c=1 float sigma r, float scale) [in] c: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*2550:ok; 1:error return: 快速双边滤波器 RGB 通道。 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] [in] 宽 char *dst, int w, int h, w:float [in] h: 高 sigma s, float [in] sigma s: 坐标空间中的滤波器西 sigma r, float scale) 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

sigma r: 颜色空间中的滤波器西

sigmaSpace 成比例。

[in]

int

void

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。 int src: 输入图像, RGB 通道 fast_bilateral_filter_color([in] [in/out] dst: 输出图像, RGB 通道 unsigned char *src, unsigned char *dst, int w, int h, int [in] 宽 w: float sigma s, float [in] h: 高 图像通道,仅 c=3 sigma_r, float _scale) c:[in] [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 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 快速双边滤波器。 int FastBilateralFilter (unsigned 输入图像 [in] src: *src, unsigned [in] guidance: 引导图像,单个通 char char 道,只有单个通道有效 *guidance, unsigned char int w, int h, int c, [in/out] dst: 输出图像 *dst, [in] 宽 float sigma s, float w: sigma r, float scale) [in] h: 高 图像通道,仅 c=1 或 c=3 [in] c: [in] sigma s: 坐标空间中的滤波 器西格玛。参数的值越大,意味着只要颜色 足够接近,更远的像素就会相互影响(请参 见 sigmaColor)。当 d>0 时,它指定邻域大 小,而不考虑 sigmaSpace。否则,d 与 sigmaSpace 成比例。 [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
	_scare = 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,	
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	Manufacture of the property of
preserve)	
void EmbossFilter(char*	浮雕过滤器,参考: preserve=1。
input, char* output, int	打网ECONOMIC 9~9: preserve 1。
preserve)	
void SharpenFilter(char*	锐化过滤器,参考: preserve=1。
•	ルルは伽柏,多写: preserve-1。
input, char* output, int	
preserve)	券和
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	THE STATE OF THE S
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage(char*	Production of the state of the
input, char* output, float	
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage1(char*	14377 A. 3 3 4 1 2 3 4
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	3
sigmal, float sigma2)	
void SkinSmooth(char*	皮肤细滑, a 是平滑级别, b 代表是否应用
input, char* output, int a, int	皮肤过滤器, a=2, b=1。
b)	
void Resizel(char*	图像模糊,w=713,h=467。
input, char* output, int w, int	
h)	
void Resize2(char*	图像模糊。
input, char* output, int w, int	
h)	
void Shift(char* input, char*	Shift函数, ch=1, v=0.1。
output, int ch, float v)	
void RGBtoHSV(char*	RGB 转 HSV。
input, char* output)	
void HSVtoRGB(char*	HSV 转 RGB。
input, char* output)	
void RGBtoLCH(char*	RGB 转 LCH。
input, char* output)	
void LCHtoRGB(char*	LCH 转 RGB。
input, char* output)	
void 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)	大克 <u>阿</u> 夫之阿 <i>内佐</i> //
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
IIShruph, Inc IIshrup),	值,如 50 50 300 300)。支持 24 位 BMP 图
	像。
void	图像层算法。
ImageLayerAlgorithm(char*	
input, char* output)	图换工 LUT 的左连从 : no.ut 且於 \ 文州
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)	图像左耳取为左连供,一旦检入文件
void RGBtoGraywithLUT(char*	图像有 LUT 的灰度化, input 是输入文件
input, char* output)	名,output 是输出文件名。支持 24 位 BMP
	图像。
void	分段线性变换, input 是输入文件名,
PiecewiseLinearTransform(cha	output 是输出文件名。支持 8 位 BMP 图像。
r* input, char* output)	
void PowerConvertion(char*	功率转换,input 是输入文件名,output 是
input, char* output, double	输出文件名。如: c = 1.2, g = 0.5。支持
c, double g)	8 位 BMP 图像。
void	拉普拉斯图像增强, input 是输入文件名,
LaplacianEnhancement(char*	output 是输出文件名。如: N=1。支持 8 位
input, char* output, int N, int	BMP 图像。
Lap1Mask[3][3])	参考模板:
	int LaplMask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void Smooth(char*	平滑, input 是输入文件名, output 是输出
input, char* output)	文件名。支持8位BMP图像。
input, char · output/	シロ.ロ。 ▽11 ○ □ □ □ □ □ □ □ □

void LaplaceSmooth(char*	拉普拉斯平滑, input 是输入文件名,
input, char* output, int N, int	output 是输出文件名。如: N=1。支持 8 位
Lap1Mask[3][3])	BMP 图像。
	参考模板:
	int Lap1Mask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void Sobel1(char*	
input, char* output, int N, int	是输出文件名。如: N=1。支持 8 位 BMP 图
Sb1Mask1[3][3], int	像。
Sb1Mask2[3][3])	参考模板:
SDIMASK2[3][3])	int SblMask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
	};
void SobelSmooth(char*	Sobel 平滑,input 是输入文件名,output
input, char* output, int N, int	是输出文件名。如:N=1。支持8位BMP图
Sb1Mask1[3][3], int	像。
Sb1Mask2[3][3])	参考模板:
	int Sb1Mask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
	};
void Multiply(char*	图像倍增化, input 是输入文件名, output
input, char* output, int N, int	是输出文件名。如: N=1。支持 8 位 BMP 图
Sb1Mask1[3][3], int	像。
Sb1Mask2[3][3], int	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
                                  };
                              图像添加, input 是输入文件名, output 是
void Add(char* input, char*
                              输出文件名。如: N=1。支持 8 位 BMP 图像。
output, int
                       N, int
Sb1Mask1[3][3], int
                               参考模板:
                               int LaplMask[3][3] = {
Sb1Mask2[3][3], int
Lap1Mask[3][3])
                                               0, 1, 0,
                                                1, -4, 1,
                                                0, 1, 0
                                  };
                                  int Sb1Mask1[3][3] = {
                                               -1, -2, -1,
                                                0, 0, 0,
                                                1, 2, 1
                                  };
                                  int Sb1Mask2[3][3] = {
                                               -1, 0, 1,
                                               -2, 0, 2,
                                               -1, 0, 1
                              功率变换, input 是输入文件名, output 是
void
     PowerConvertion1(char*
                              输出文件名。如: c = 1.2, g = 0.5, N=1。
input, char*
               output, double
                               支持 8 位 BMP 图像。
c, double
             g, int
                       N, int
Sb1Mask1[3][3], int
                               int LaplMask[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
	};
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	怀旧滤镜,支持 24 位 BMP 图像。
NostalgicFilter(BMPMat**	
input,BMPMat** output)	
void	图像放缩,支持8位BMP图像。
SizeTransformation(short**	
input, short** output, short	
height, short width, short	
out_height, short out_width)	
void ReverseColor(short**	图像反色。
input, short** output, long	
height, long width, short	
GRAY_LEVELS)	
void Logarithm(short**	对数变换,默认 c=10。
input, short** output, long	
height, long width, short c)	
void Gamma(short**	幂律(伽马)变换,默认 c=1.2。
input, short** output, long	
height, long width, double c)	
void	直方图均衡化。
HistogramEqualization(short*	

* input, short** output, long	
height, long width, short	
GRAY_LEVELS)	
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	
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	
sob1e2[3][3])	
<pre>void DFTRead(short** input,</pre>	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**	13. ATT INVITED NEV
input, short** output, long	
height, long width)	
void MeanFilter(short**	均值滤波器。
input, short** output, long	~~
height, long width)	
nergitt, rong wrutii/	

void	几何均值滤波器,默认 product=1.0。
GeometricMeanFilter(short**	
input, short** output, long	
height, long width, double	
product)	MENTAL MANENTAL BINAL O
void	谐波均值滤波,默认 sum=0。
HarmonicMeanFiltering(short*	
* input, short** output, long	
height, long width, double	
sum)	WARRAL II. III. III. III. III. III. III. III
void	逆谐波均值滤波,Q为滤波器的阶数,Q为
InverseHarmonicMeanFiltering	正时,消除胡椒噪声,Q为负时消除盐粒噪
(short** input, short**	声, Q=0 为算术均值滤波器, Q=-1 谐波均值
output, long height, long	滤波器,默认 Q=2。
width, int Q)	
void Threshold(short**	基本全局阈值处理方法。
input, short** output, long	
height, long width, int	
delt_t, double T)	
void OTSU(short**	Otsu 方法进行最佳全局阈值处理。
input, short** output, long	
height, long width, short	
GRAY_LEVELS)	
void	基于模板矩阵的全局相加。
MatrixGlobalAddition24(BMPMa	
t** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相减。
MatrixGlobalSubtraction24(BM	
PMat** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相乘。
MatrixGlobalMultiplication24	
(BMPMat** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相除。
MatrixGlobalDivision24(BMPMa	
t** input1, BMPMat**	
input2, BMPMat** output)	
void	基于模板矩阵的全局相加。
MatrixGlobalAddition32(BMPMa	
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	至 1 疾似尼叶的主用相称。
input2, BMPMat** output)	+ - + + + + + + + + + + + + + + + + + +
void	基于模板矩阵的全局相加。
MatrixGlobalAddition8(unsign	
ed char** input1, unsigned	
char** input2, unsigned	
char** output)	
void	基于模板矩阵的全局相减。
MatrixGlobalSubtraction8(uns	
igned char** input1, unsigned	
char** input2, unsigned	
char** output)	
void	基于模板矩阵的全局相乘。
MatrixGlobalMultiplication8(
unsigned char**	
input1, unsigned char**	
input2, unsigned char**	
output)	
void	基于模板矩阵的全局相除。
MatrixGlobalDivision8(unsign	至1保仅作时主用相协。
ed char** input1, unsigned	
char** input2, unsigned	
char** output)	业人园以供业子子口却共历日井户部八届
void	彩色图以矩形方式局部截取且其它部分填
ColorRectangleLocalSegmentat	充,(x1, y1)是矩形的左上角的坐标,
ion(char* input, char*	(x2, y2)是矩形的右下角的坐标。
output, int x1, int y1, int	函数源代码:
x2, int y2, BMPMat color)	需引入以下头文件:
	typedef struct {
	unsigned char B;
	unsigned char G;
	unsigned char R;
	unsigned char A;
	}BMPMat;
	声明:

```
unsigned
                                                        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:
                                    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
                                                               i
                                                      int
                               0; i \leq height; i++) {
                                        for (unsigned
                                                         int
                                                                 j
                               0; j \le dth; j++) {
                                           output[i][j]. B=color. B;
                                           output[i][j]. G=color. G;
                                           output[i][j]. R=color. R;
                                    for (unsigned
                                                      int
                               y1; i = y2; i++  {
                                        for (unsigned
                                                         int
                                                                 j
                               x1; j \le x2; j++) {
                               output[i][j].B=input[i][j].B;
                               output[i][j].G=input[i][j].G;
                               output[i][j]. R=input[i][j]. R;
                               GenerateImage (outputfile, output, 24);
                                灰度图以矩形方式局部截取且其它部分填
void
```

```
充,(x1,y1)是矩形的左上角的坐标,
GrayRectangleLocalSegmentati
                               (x2, y2) 是矩形的右下角的坐标。
on (char*
                  input, char*
                       y1, int
                               函数源代码:
output, int
             x1, int
                               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:
                                                      BMPRead8(char*
                               unsigned
                                           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
                               y1;i = y2;i++) {
                                       for (unsigned
                                                       int
                                                               j
                               x1; j \le x2; j++) {
                                           output[i][j]=input[i][j];
```

```
GenerateImage8(outputfile, output);
                               彩色图画矩形,(x1, y1)是矩形的左上角
void
ColorDrawRectangle(char*
                              的坐标,(x2, y2)是矩形的右下角的坐标。
                              函数源代码:
input, char*
                  output, int
                              需引入以下头文件:
x1, int
                      x2, int
           y1, 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
                                                              j
                              0; j \le dth; j++) {
                                         output[i][j]. B=color. B;
                                         output[i][j]. G=color. G;
                                         output[i][j]. R=color. R;
                                   for (unsigned
                                                    int
```

```
0; i \leq 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);
                               灰度图画矩形,(x1, y1)是矩形的左上角
void GrayDrawRectangle(char*
                               的坐标,(x2, y2)是矩形的右下角的坐标。
input, char*
                   output, int
x1, int
           y1, int
                       x2, int
                               函数源代码:
                               需引入以下头文件:
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);
                   GenerateImage(char*
void
output, BMPMat** color, unsigned short
type);
unsigned int BMPHeight(char* input);
unsigned int BMPWidth(char* input);
参考例程:
    unsigned char color=255;
    unsigned
                                 char**
input=BMPRead8(inputfile);
    unsigned
                                 char**
output=BMPRead8(inputfile);
    unsigned
                                     int
height=BMPHeight(inputfile);
    unsigned
                                     int
width=BMPWidth(inputfile);
    for (unsigned
                       int
                                i
0; i < height; i++) {
        for (unsigned
                          int
                                 j
0; j \le dth; j++) {
           output[i][j]=color;
    for (unsigned
                                i
                       int
0; i \leq height; i++) {
        for (unsigned
                          int
                                 j
0; j \leq width; j++) \{
           if(j)=x1&&j<=x2&&i==y1)
             output[i][j]=color;
             if(j==x1\&\&i>=y1\&\&i<=y2)
             output[i][j]=color;
             if(j==x2\&\&i>=y1\&\&i<=y2)
             output[i][j]=color;
             if(j)=x1&&j<=x2&&i==y2
             output[i][j]=color;
```

	}
	}
	}
	GenerateImage8(outputfile, output);
void Relief(BMPMat**	浮雕效果,默认 value=128。
input, BMPMat** output, int	
value)	
void Relief(unsigned char**	浮雕效果,默认 value=128。
input, unsigned char**	
output, int value)	ET In IVI II III III II
void Sharpening (BMPMat**	图像锐化,默认 degree=0.3。
input, BMPMat** output, double	
degree)	FFI Its, Ist II. While the second
void Sharpening (unsigned	图像锐化,默认 degree=0.3。
char** input, unsigned char**	
output, double degree)	园协学儿、 图以 1 0
void Soften (BMPMat**	图像柔化,默认 value=9。
input, BMPMat** output, int	
value)	历 <i>协</i> 之儿 图NI 1 0
void Soften (unsigned char**	图像柔化,默认 value=9。
input, unsigned char**	
output, int value)	V 之台和## 士持 TDC 立伊
<pre>void flipX(char* input, char*</pre>	A 刀凹翻转,又持 JPG 又件。
<pre>output) void flipY(char* input, char*</pre>	Y 方向翻转,支持 JPG 文件。
output)	I 刀凹翻投,又付 JI G 又什。
void Crop(char* input, char*	裁剪。
output, uint16 t start x,	88,930
uint16_t start_y, uint16_t	
new_height, uint16_t	
new_merght, unitio_t new_width)	
void Resize(char*	
input, char* output, int	10/2 **
new_width, int new_height)	
void Scale(char* input, char*	比例。
output, double ratio)	
void GrayscaleAvg(char*	灰度平均值。
input, char* output)	
void grayscaleLum(char*	灰度亮度。
input, char* output)	
void ColorMask(char*	彩色遮罩。
input, char* output, float	
r,float g,float b)	

void PixeLize(char*	像素化,参考: strength=2。
input, char* output, int	家来記,多項·Strength 2。
strength)	
void GaussianBlur(char*	高斯模糊,参考: strength=2。
input, char* output, int	同が伝統。多句·Strength 2。
strength)	
void EdgeDetection(char*	边缘检测,参考: cutoff=115。
input, char* output, double	过场位例,参与: Cutoff-115。
cutoff)	
void Sharpen(char*	锐化。
input, char* output)	թե ru •
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)	1 ~3/0\/X ru o
void SimpleBW(char*	简易 BW。
input, char* output)	
void AdvancedBW(char*	高级 BW。
input, char* output)	H-13/X D11 0
void UniformNoise(char*	均匀噪声。
input, char* output)	
void GaussianNoise(char*	高斯噪声。
input, char* output, double	1-4791 XK7 V
sigma)	
void	椒盐噪声。
SaltAndPepperNoise(char*	7107
input, char* output)	
void MeanFilter(char*	均值滤波。
input, char* output, int	
filterSize)	
void GaussianFilter(char*	高斯滤波。
input, char* output, double	
sigma)	
void MedianFilter(char*	中值滤波。
input, char* output, int size)	
void	有效均值滤波器。
EfficientMeanFilter(char*	
input, char* output, int	
filterSize)	
double	均方误差, 计算图像相似度, 返回值越小图
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)	The XIII of Edition of the State of the Stat
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)	
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)	/X14 °
void	直方图均衡化。
HistogramEqualization8(char*	
input, char* output)	
void Smooth(char*	平滑。
input, char* output)	1140
void CannyEdge(char*	Canny 算子。
input, char* output)	333333
void EdgeEnhance(char*	边缘增强。
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
input, char* output)	名。支持 8 位 BMP 图像。
void TextTest(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持 8 位 BMP 图像。
1,	- 744 - III - III III III III

void RedChannel(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。
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 HistogramStatistics(char* input, char* output)	直方图统计, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。
<pre>void HistogramEqualization1(char* input, char* output)</pre>	直方图均衡化, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。
<pre>void ReflectionRay(char* input, char* output) void MeanFiltering24(char*</pre>	反射线, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。 均值滤波, input 是输入文件名, output 是
<pre>input, char* output) void MedianFiltering24(char* input, char* output)</pre>	输出文件名。支持 24 位 BMP 图像。 中值滤波, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。
void ZoomOutAndZoomIn(char* input, char* output, double value)	缩放(双线性插值), input 是输入文件名, output 是输出文件名。value 是放大倍数, 如 value=0.5。支持24位BMP图像。
void Translation24(char* input, char* output, int x, int y)	平移, input 是输入文件名, output 是输出文件名。x 是横轴的平移量, y 是纵轴的平移量, 如 x=-10, y=-30。支持 24 位 BMP 图像。
void Mirror24(char* input, char* output)	镜像, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。
void Rotate24(char* input, char* output, double degree)	旋转, input 是输入文件名, output 是输出文件名。degree 是旋转的度数。支持 24 位 BMP 图像。
void GivenThresholdMethod(char* input, char* output, int threshold)	给定阈值法处理图像,使图片黑白化,input 是输入文件名,output 是输出文件名。threshold 是 给 定 的 阈 值 , 如 threshold=100。支持 24 位 BMP 图像。
void IterativeThresholdMethod(cha r* input, char* output)	迭代阈值法处理图像,使图片黑白化,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。
void OstuThresholdSegmentationMet hod(char* input, char* output)	Ostu (大津法) 阈值分割, input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。

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
	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
Boutput, char Boutput,	24 位 BMP 图像。
void ReverseColor(char*	对灰度图进行反色, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
Image1* LoadImage1(char*	BMP 图像读取, input 是输入文件名。支持
input)	8 位和 24 位 BMP 图像。
	返回 Image1 型数据,Image1 型数据的结构
	如下:
	typedef struct
	{
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
	}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:
	typedef struct
	{
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
. ,	Image1;
void	图像对比度扩展, input 是输入文件名,
ImageContrastExtension(char*	output 是输出文件名。
input, char* output, double	其中,可参考: double
m, double g1, double g2, double	m=1.5, g1=100.0, g2=200.0; m 对应斜率
a)	double $a=(255.0-m*(g2-g1))/(255.0-$
	(g2-g1)); 去转 g 台 DMD 図 梅
	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
threshord)	位 BMP 图像。
void	全局二值化,input 是输入文件名,output
GlobalBinarization(char*	主周二值化,Imput 定揃八叉[F右,output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	定制山文作石。文持 O 位 DMF 图像。
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)	output 定制山文什石。文持 o 应 DMF 图像。
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)	制山文件石。文持 0 世 DMF 图像。
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)	棚山人下石。又打 0 世 DML 営隊。
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	
input, unsigned int newWidth,	
unsigned int newHeight)	
IMAGE	缩放图片(双线性插值法)。
TransformShapeLinear(IMAGE	
input, unsigned int newWidth,	
unsigned int newHeight)	
IMAGE	图像的任意角度的旋转。
TransformShapeWhirl(IMAGE	
1 - \	1

input, float angle)	
IMAGE	图像的镜像翻转。
TransformShapeUpturn(IMAGE	H WHY OU DY MAN IN C
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
E III, III Grayscaremode)	量法,6表示蓝色分量法。
	二值图(自定义阈值法)。
void	—但囟(日疋又阕但法)。
TransformColorBWDIY(IMAGE	
input, unsigned char	
Threshold)	- H E / L H H A A B H T T T T T T T T T T T T T T T T T T
void	二值图(大津法 OSTU,适用双峰直方图。)
TransformColorBWOSTU(IMAGE	
input)	
void	二值图(三角法 TRIANGLE,适用单峰直方
TransformColorBWTRIANGLE(IMA	图。)
GE input)	
IMAGE	二值图(自适应阈值法, areaSize=25 较合
TransformColorBWAdaptive(IMA	适)
GE input, int areaSize)	
IMAGE	二值图(用二值图表示灰度变
TransformColorBWGrayscale(IM	化, areaSize=25 较合适)
AGE input, int areaSize)	
void	反色。
TransformColorOpposite(IMAGE	
input)	
IMAGE	直方图均衡化(分步计算,效果更加柔和)。
TransformColorHistogramPart(
IMAGE input)	
IMAGE	直方图均衡化(整体计算,效果更加尖锐)。
TransformColorHistogramAll(I	
MAGE input)	
IMAGE KernelsUseDIY(IMAGE	卷积操作(自定义)。
input, double* kernels, int	
areaSize, double modulus)	
IMAGE	中值滤波。
WavefilteringMedian(IMAGE	
input)	
IMAGE	高斯滤波。
WavefilteringGauss(IMAGE	高斯滤波卷积核:
input, double	double KERNELS_Wave_Gauss[9] =
KERNELS_Wave_Gauss[9], int	{
a, double b)	1, 2, 1,
	1

```
2, 4, 2,
                                 1, 2,1
                             };
                             低通滤波。
IMAGE
Wavefiltering LowPass(IMAGE
                             // 低通滤波卷积核 LP1
input, double* kernels)
                             double KERNELS Wave LowPass LP1[9] =
                                 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。
                             // 高通滤波卷积核 HP1
WavefilteringHighPass(IMAGE
                             double KERNELS Wave HighPass HP1[9] =
input, double* kernels)
                                 -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
                            均值滤波。
IMAGE
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
IMAGE
                            差分边缘检测。
                            // 差分垂直边缘检测卷积核
EdgeDetectionDifference(IMAG
E input, double* kernels)
                            doub1e
                            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
```

```
Sobel 边缘检测。
IMAGE
                             // Sobel X边缘检测卷积核
KernelsUseEdgeSobel(IMAGE
input,
        double*
                  kernels1,
                             double KERNELS Edge Sobel X[9] =
double* kernels2)
                                 -1, 0, 1,
                                - 2, 0, 2,
                                 -1, 0, 1
                             };
                             // Sobel Y边缘检测卷积核
                             double KERNELS_Edge_Sobel_Y[9] =
                                -1, -2, -1,
                                0, 0, 0,
                                 1, 2, 1
IMAGE
                             Laplace 边缘检测。
                             // Laplace 边缘检测卷积核 LAP1
EdgeDetectionLaplace(IMAGE
input, double* kernels)
                             double KERNELS Edge Laplace LAP1[9] =
                                 0, 1, 0,
                                 1, -4, 1,
                                 0, 1, 0
                             };
                             // Laplace 边缘检测卷积核 LAP2
                             double KERNELS_Edge_Laplace_LAP2[9] =
                                -1, -1, -1,
                                -1, 8, -1,
                                -1, -1, -1
                             };
                             // Laplace 边缘检测卷积核 LAP3
                             double KERNELS Edge Laplace LAP3[9] =
                                -1, -1, -1,
                                -1, 9, -1,
                                -1, -1, -1
                             };
                             // Laplace 边缘检测卷积核 LAP4
                             double KERNELS_Edge_Laplace_LAP4[9] =
```

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

	DODA . 1
	BGRA* color;
	}IMAGE, *PIMAGE;
	声明:
	<pre>IMAGE Image_bmp_load(char* filename);</pre>
	void Image_bmp_save(char*
	filename, IMAGE im);
	参考:
	// 用于处理
	IMAGE input2 =
	Image_bmp_load(inputfile); // 用于保存
	IMAGE input2=
	Image_bmp_load(inputfile);
	input2=FaceDetection(input1, input2, KE
	RNELS Wave Average);
	// 保存图片
	Image_bmp_save(outputfile, input2);
void	图像积分图。
IntegralDiagram(unsigned int	
*input, unsigned int *output,	
int width, int height)	
void ImageEncryption(char*	图像加密,支持8位、24位和32位BMP图
inFileName, char*	像。inFileName 是原图图像文件名,
outFileName, char key)	outFileName 是解密图像文件名, key 是密
, , , , , , , , , , , , , , , , , , , ,	钥,如 key=255。
void ImageDecryption(char*	图像解密, in File Name 是加密图像文件名,
inFileName, char*	outFileName 是解密图像文件名,key 是密
outFileName, char key)	钥,如 key=255。支持 8 位、24 位和 32 位
, , , , , ,	BMP 图像。
void Compress8(string	图像压缩, input 是输入文件名, output 是
input, string output)	输出文件名。支持 8 位 BMP 图像。
void Decompression(string	图像解压, input 是输入文件名, output 是
input, string output)	输出文件名。支持 8 位 BMP 图像压缩后的结
	果文件。
void HorizontalMirror(char*	水平镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void MirrorVertically(char*	垂直镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void XMirroring(char*	X 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void YMirroring(char*	Y 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void ImageConvolution(char*	图像卷积, input 是输入文件名, output 是
input, char* output, double**	输出文件名。Kernel 是卷积核,如 double

V1 :+)	V1[2][2] - [[0 995 0 995
Kernel, int n, int m)	Kernel[3][3] = {{-0.225, -0.225-
	0. 225}, {-0. 225, 1, -0. 225}, {-0. 225, -
	0.225,-0.225}}; n 是 Kernel 的第一维的
	大小,m 是 Kernel 的第二维的大小,形如
	Kernel[n][m]。支持 24 位 BMP 图像。
void SpatialMeanFiter(char*	空间均值过滤器,参考: radius=3。
input, char* output, int	
radius)	
void	空间中值过滤器,参考: radius=3。
SpatialMedianFiter(char*	
input, char* output, int	
radius)	
void SpatialMaxFiter(char*	空间最大过滤器,参考: radius=3。
input, char* output, int	
radius)	
void SpatialMinFiter(char*	空间最小过滤器,参考: radius=3。
input, char* output, int	Thom, but it is a second of the second of th
radius)	
void SpatialGaussFiter(char*	空间高斯过滤器,参考: radius=3。
input, char* output, int	工門同別及心部,多写:Tautus-5。
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 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)	6.10
void STDFT1(char*	参考: inv=false。
input, char* output, bool inv)	
void STDFT2(char*	参考: inv=false。
input, char* output, bool inv)	
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	1 / 1 / 1 / 1 / 1 / 1
Tipac, Dili liac Odepac, Dili liac	

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	
<u> </u>	
target)	图像上泛 (-1 -1) 具短形泛法区的七上角
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)	图像本是 / 1 1) 目标取得速度处土工力
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	
1x, float 1y)	
,	

void	双线性插值法去栅格, input 是输入文件
ImageScalingBilinearInterpol	名, output 是输出文件名。lx 和 ly 是长和
ation(char* input, char*	宽需要缩放的倍数。支持 8 位 BMP 图像。
output, float 1x, float 1y)	
void	双线性插值,input 是输入文件名,output
BilinearInterpolationScaling	是输出文件名。ExpScalValue 是期望的缩
(char* input, char*	放倍数 (允许小数)。支持 BMP 图像。
output, float ExpScalValue)	
void	最近邻插值,input 是输入文件名,output
NearestNeighborInterpolation	是输出文件名。ExpScalValue 是期望的缩
Scaling(char* input, char*	放倍数 (允许小数)。支持 BMP 图像。
output, float ExpScalValue)	
void ZoomImg(unsigned char	二次线性插值图像缩放。
*input, unsigned char	
*output, int sw, int sh, int	
channels, int dw, int dh)	
void	图像修复, output 用于保存结果 (与 input
CrossDenoising24(BMPMat**	大小相同), target 是污点像素, weight 是
input,BMPMat** output,BMPMat	修复权重系数。
target, BMPMatdouble weight)	
void	图像修复, output 用于保存结果(与 input
CrossDenoising8(unsigned	大小相同), target 是污点像素, weight 是
char** input, unsigned char**	修复权重系数。
output, unsigned char	
target, double weight)	
void	input 是输入文件名, output 是输出文件
RotateRight90Degrees(char*	名。支持 8 位 BMP 图像,向右旋转 90 度。
input, char* output)	
void	input 是输入文件名, output 是输出文件
RotateLeft90Degrees(char*	名。支持8位BMP图像,向左旋转90度。
input, char* output)	
void ImageRotation(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 8 位 BMP 图像。angle 是
angle)	要旋转的角度。
void Rotation8(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 8 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标, color 是旋转后非原
color)	图区域的填充颜色。
void Rotation24(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 24 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标; red、green、blue 分
red, unsigned char	别是旋转后非原图区域要填充的颜色的红

green, unsigned char blue)	绿蓝分量。
void Rotation(char*	图像旋转, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。angle 是
angle, unsigned char color)	旋转的角度, color 是旋转后非原图区域填
	充的颜色,如 color=100。
void Rotate(char*	图像旋转, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 BMP 图像。angle 是旋转
angle)	的角度。
void	灰度图像旋转 90。
imgRotate90Gray(unsigned	
char *input,unsigned char	
*output, int sw, int sh, int	
*dw, int *dh)	
void	彩色图像旋转 90。
imgRotate90Color(unsigned	
char *input,unsigned char	
*output, int sw, int sh, int	
*dw, int *dh)	
void	灰阶图像旋转 270。
imgRotate270Gray(unsigned	
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	- 114 14 XIC/ / > J. G. G. O. D. BOOO
zarpat, onar output, riout	

angle, int interpolation)	
void HSV (char* input, char*	图梅布调炼和度明度调带 <u>会</u> 老 h-190
	图像色调饱和度明度调节,参考: h=120,
output, int h, int s, int v)	s=60, v=20。 - 新 <i>朱</i>
void ColorTransfer1(char*	颜色转移,支持 BMP 图像。
input1, char* input2, char*	
output)	
void OilpaintFilter(char*	油画滤镜,参考: radius=10, smooth=100。
input, char* output, int	
radius, int smooth)	
void HaloFilter(char*	晕角滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void GrayHistogram(char*	灰度直方图, 参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void RedHistogram(char*	红色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void GreenHistogram(char*	绿色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void BlueHistogram(char*	蓝色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization2(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output, int	BMP 图像。imgBit 是输入图像的位数。
imgBit)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization3(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output)	BMP 图像。
void	直方图均衡化, input 是输入文件名,
HistogramEqualization4(char*	output 是输出文件名。支持 8 位和 24 位
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,
HistogramEqualization(char*	hHeight=100.
input, char* output, int	
hWidth, int hHeight)	
void	灰度直方图,参考: hWidth=256,
GrayHistogramEqualization(ch	hHeight=100.
ar* input, char* output, int	
hWidth, int hHeight)	
void	红色通道直方图,参考: hWidth=256,
1014	五日200 日7日,乡 7· IIII 1011 200;

RedHistogramEqualization(cha	hHeight=100。
r* input, char* output, int	
hWidth, int hHeight)	
void	绿色通道直方图,参考: hWidth=256,
GreenHistogramEqualization(c	hHeight=100。
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)	
void	蓝色通道直方图,参考: hWidth=256,
BlueHistagramStretch(char*	hHeight=100。
input, char* output, int	
hWidth, int hHeight)	
void MedianFilteringl(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持8位BMP图像。
void MedianFiltering2(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
void	阈值处理, input 是输入文件名, output 是
ThresholdProcessing(char*	输出文件名。支持 8 位 BMP 图像。Threshold
input, char* output, int	是阈值相关参数,如 Threshold=0.001。
Threshold)	
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)	机固角状剂 取取固体 十叶 吸鱼 取鱼
void ToRIM(char* input, char*	一般图像转到 RIM 图像,支持 PNG、JPG 和

output)	TGA 图像。
void ToImage(char*	RIM 图像转到一般图像,支持 PNG、JPG 和
input, char* output, int	TGA 图像。 jpg_quality=25。
<pre>jpg_quality)</pre>	
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 ESC * 指
skip_cmd, unsigned short	\$.
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。支持 BMP 图像。
char* input, const char*	
output)	

其他处理

void Encode(char* input, char*	文本文件压缩, input 是输入文件名,
output)	output 是输出文件名。
void Decode(char* input, char*	文本文件压缩结果解压缩, input 是
output)	输入文件名,output 是输出文件名。
void FileCompress(char *input ,	文件压缩, input 是输入文件名,
char *output)	output 是输出文件名。
void FileDecompression(char	文件压缩结果解压缩, input 是输入
*input , char *output)	文件名,output 是输出文件名。