#### 使用说明书

# 目录

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

#### PPM、PGM 和 PBM 图像处理

OTSU 二值化。input 是输入文件名,
output 是输出文件名。支持 P5 格式的
PGM 图像。
PPM 图像转为 BMP 图像。bpp 是 BMP 图
像的色深。
BMP 图像转 PPM 图像。
PPM 图像转为 BMP 图像。bpp 是 BMP 图
像的色深。
BMP 图像转 PPM 图像。
BMP 转 PGM。
BMP 转 PPM。
PPM 转 PGM。
PPM 图像滤波。
PGM 图像滤波。
OTSU二值化划分。input 是输入文件名,
output 是输出文件名。支持 P5 格式的
PGM 图像。
PPM 图像模糊, input 是输入文件名,
output 是输出文件名。支持 P3 格式的
PPM 图像。
读取 PBM 图像并返回图像数据。input
是要读取的 PBM 图像文件名。支持 P4

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

void SepiaFilter(char*	乌贼墨过滤器,input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P6 格式的
Thiput, char * output)	
	PPM 图像。
void AdjustSaturation(char*	调整图像饱和度,input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标饱和度,
	如 a=30。支持 P6 格式的 PPM 图像。
void Resize(char* input, char*	调整图像大小,input 是输入文件名,
output, unsigned int NewWidth,	output 是输出文件名。NewWidth 和
unsigned int NewHeight)	NewHeight 分别是输出图像的宽和高。
	支持 P6 格式的 PPM 图像。
void AdjustHue(char* input, char*	调整图像的色调, input 是输入文件名,
output, int a)	output 是输出文件名。a 是目标色调,
output, int a)	如 a=125。支持 P6 格式的 PPM 图像。
11 A1: (1	
void AdjustBrightness(char*	调整图像亮度,input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标亮度,
	如 a=60。支持 P6 格式的 PPM 图像。
void AdjustContrast(char*	调整图像对比度, input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标对比度,
	如 a=60。支持 P6 格式的 PPM 图像。
void AdjustBlur(char*	通过 sigma 因子模糊图像, input 是输
input, char* output, double a)	入文件名, output 是输出文件名。a 是
	sigma 因子,如 a=5。支持 P6 格式的 PPM
	图像。
void MeanGrayFilter(char*	平均灰度滤波器, input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是平均值系数,
	如 a=3。支持 P6 格式的 PPM 图像。
void Pixelate(char* input, char*	像素化, input 是输入文件名, output
output, unsigned int a)	是输出文件名。a 是幅度值,如 a=8。
output, unsigned int a)	支持 P6 格式的 PPM 图像。
usid Detete (sharak input sharak	
void Rotate(char* input, char*	旋转图像,input 是输入文件名,output
output, short a)	是输出文件名。a 是旋转的角度,如
,	a=45。支持 P6 格式的 PPM 图像。
void GammaCorrection(char*	伽马校正, input 是输入文件名, output
input, char* output, double a)	是输出文件名。a 是 gamma 数,如 a=0.5。
	支持 P6 格式的 PPM 图像。
void	生成灰度图以及 RGB 通道分离, input
GrayAndChannelSeparation(char*	是输入的 P6 格式的 PPM 图像;
input, char* Grayoutput, char*	Grayoutput 是输出的灰度图文件名,
Routput, char* Goutput, char*	Routput、Goutput 和 Boutput 分别是输
Boutput)	出的 R、G和 B通道的图像文件名,输
	出都是 PGM 格式文件。
void PGMBin(char* input, char*	灰度图像二值化,输入是灰度图像,输
	入和输出都是 PGM 文件, threshold 是
output, int threshold)	
	阈值,如 threshold=125。

	작/ 및 ID // 나스
void Brightening(char*	彩色图像增亮,输入和输出都是 P6 格
input, char* output, int a)	式的 PPM 图像, a 是增亮系数, 如 a=80。
void GrayBrightening(char*	灰度图像增亮,输入和输出都是 PGM 图
input, char* output, int a)	像,a 是增亮系数,如 a=80。
void PPMFilter(char* input, char*	彩色图像滤波,输入和输出都是 P6 格
output)	式的 PPM 文件。
void PGMGrayFilter(char*	灰度图像滤波,输入和输出都是 PGM 图
input, char* output)	像。
void PPMtoBMP(char* input, char*	PPM 图像转 BMP 图像, input 是输入文
output)	件名, output 是输出文件名。支持 P6
	格式的 PPM 图像。
void PGMRotated(char*	图像旋转, channels 是输入图像的通
input, char* output, int width, int	道,theta 是旋转弧度,支持 PGM 图像。
height, int channels, double	参考: theta=45.0*3.1415926/180。
theta)	3 J. the tu 10.0.0.1110020, 1000
void XCorner(char* input, char*	X 方向角度, channels 是输入图像的通
output, int width, int height, int	道。支持 PGM 图像。
channels, double theta)	
void YCorner(char* input, char*	Y 方向角度, channels 是输入图像的通
output, int width, int height, int	道。支持 PGM 图像。
	但。又将 FOM 图像。
channels, double theta)	<b>圆梅拱</b> ₩ -11- 目於 ) 圆梅的语
void Smooth(char* input, char*	图像模糊, channels 是输入图像的通
output, int width, int height, int	道, sigma_x 是 X 方向的模糊系数,
channels, float sigma_x, float	sigma_y 是 Y 方向的模糊系数。
sigma_y, double theta)	
void PGMOtsuThreshold(string	大津阈值法, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P5 格式的
. ,	PGM 图像。
void	局部大津阈值,input 是输入文件名,
PGMLocalisedOtsuThreshold(string	output 是输出文件名。支持 P5 格式的
input, char* output)	PGM 图像。
void PGMSauvolaThreshold(string	索沃拉阈值,支持 P5 格式的 PGM 图像。
input, char* output, double	a、b和c的参考值如:
a, double b, double c)	a=0. 01, b=15, c=225.
void PGMThreshold(string	阈值法, input 是输入文件名, output
input, char* output, int thresh)	是输出文件名。支持 P5 格式的 PGM 图
	像。thresh 是阈值,如:thresh=5。
float Repair1(char* input, char*	图像修复, var 是噪声方差, threshold
output, float var, float	是阈值, nbLevels 是要处理的级别数,
threshold, int nbLevels, float a)	a=10。返回 ISNR。
float Repair2(char* input, char*	图像修复, var 是噪声方差, threshold
output, float var, float	是阈值, nbLevels 是要处理的级别数,
threshold, int nbLevels, float a)	a=10。返回 ISNR。
void LowPassFilterRepair1(char*	低通滤波图像修复, a=10, b=6,

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

## YUV 图像处理

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

height, unsigned char Y_BLACK, unsigned char	原图中的黑色变成透明,参
U_BLACK, unsigned char V_BLACK)	考: Y_BLACK=16,
	U_BLACK=128 ,
	V_BLACK=128。
void YUVsuperposition(char* input1, char*	yuv444p 直接叠加到
input2, char* output, int width, int	yuv420p 上,不做转换,
height, unsigned char Y_BLACK, unsigned char	Y_BLACK 、 U_BLACK 和
U_BLACK, unsigned char V_BLACK)	V_BLACK 用于将原图中的黑
	色变成透明,参考:
	Y_BLACK=16, U_BLACK=128,
	V_BLACK=128。
void YUV444toYUV420(char* input, char*	YUV444 转 YUV420, height
output, int height, int width)	是输入的YUV444文件的高,
	width 是输入的 YUV444 文
	件的宽。
void YUV444toYUV420(char* input,char*	YUV444 转 YUV420, height
output, int height, int width, int frames)	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*	YUV444 转到 YUV420 上的叠
input2, char* output, int width, int	加,Y_BLACK、U_BLACK 和
height, unsigned char Y_BLACK, unsigned char	V_BLACK 用于将原图中的黑
U_BLACK, unsigned char V_BLACK)	色变成透明,参考:
	Y_BLACK=16, U_BLACK=128,
	V_BLACK=128。
<pre>void YUVEdgeProcessingY(char* input, char*</pre>	YUV 边缘处理, input 是输
output, int width, int height, double k)	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*</pre>	YUV 边缘处理, input 是输
output, int width, int height, double k)	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*</pre>	YUV 边缘处理, input 是输
output, int width, int height, double k)	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*</pre>	YUV 加载 BMP, inputBMP 是
inputYUV,char* output,int YUVwidth,int	输入的 BMP 图像, inputYUV
YUVheight, int depth, bool mt)	是输入的 YUV 图像,
	inputYUV 起到容器的作用,
inputYUV,char* output,int YUVwidth,int	k=0.5。 YUV 加载 BMP, inputBMP 是输入的 BMP 图像,inputYUV 是输入的 YUV 图像,

	1
	YUVwidth 和 YUVheight 是
	输入的 YUV 图像的宽和高,
	参考: depth=12, mt=true。
void	YUV 仅水平方向的边缘处
YUVEdgeProcessingHorizontalDirection(char*	理, input 是输入文件名,
input, char* output, int width, int	output 是输出文件名。
height, double k)	width 和 height 是输入图
	像的宽和高。参考: k=0.7。
void YUVVieoEdgeProcessing(char*	YUV 视频文件边缘处理,
input, char* output, int width, int	input 是输入文件名,
height, int frame, int max_frame)	output 是输出文件名。
	width 和 height 是输入图
	像的宽和高,frame 是要处
	理的帧序号,max_frame 是
	最大帧序号。
void YUVScale(char* input, char* output, int	缩放 yuv420 图像,参考:
inputWidth, int inputHeight, int	inputWidth=1280 ,
outputWidth,int outputHeight)	inputHeight=720 ,
	outputWidth=128 ,
	outputHeight=72。
<pre>void NoiseTreatment(char* input, char*</pre>	YUV 噪声处理。
output, int width, int height, int	
TWICEwidth, int TWICEheight)	
<pre>void NoiseTreatment(char* input, char*</pre>	YUV 噪声处理。
output, int width, int height, int frame, int	
max_frame)	

## RAW 图像处理

unsigned char** RAWRead(char*	读取 RAW 图像。
input, int height, int width)	
void RAWWrite(unsigned char**	保存 RAW 图像。
input, char* output, int height, int	
width)	
void MBVQ(char* input, char*	MBVQ 效果, input 是输入文件名,
output, int width, int height)	output 是输出文件名。width 和
	height 是输出图像的宽和高。
void RAWtoPPM_red(char*	RAW 转为 PPM 后提取红色通道,参考:
input, char* output, int width, int	width=4096 , height=3072 ,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或 LINEAR。
	支持 RAW12 格式。
	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR

	}:
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。
neight, bebayernigorithm argo,	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:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
void RAWtoPPM blue(char*	RAW 转为 PPM 后提取蓝色通道,参考:
input, char* output, int width, int	width=4096 , height=3072 ,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或 LINEAR。
	支持 RAW12 格式。
	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	} .
void RAWtoPPM(char* input, char*	RAW 转为 PPM,参考: width=4096,
output, int width, int	height=3072
	algo=NEARESTNEIGHBOUR 或 LINEAR。
height, DebayerAlgorithm algo)	支持 RAW12 格式。
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
.1. D. D	写好变物
void RawPowerTransformation(char*	幂次变换,input 是输入的 RAW 图像
input, char* output, int width, int	文件名, output 是输出的 RAW 图像文
height, int c, float v)	件名,width 是输入图像的宽,height
	是输入图像的高。默认 c=1, v=0.6。
	支持 RAW 图像。
void RAWAvgFilter(char*	平均滤波器, input 是输入文件名,
input, char* output, int ROWS, int	output 是输出文件名。ROWS 是图像

COLS, int M, float mask[3][3])	的行大小,COLS 是图像的列大小,M 是滤波相关参数,如 M=1; mask 是滤 波器模板。支持 RAW 图像。 参考模板: float mask[3][3] = {{0.1111,0.1111,0.1111},
	$\{0.1111, 0.1111, 0.1111\}\};$
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
	是输入图像的高。支持 RAW 图像。
void	直方图均衡化, input 是输入的 RAW
RawHistogramEqualization(char*	图像文件名, output 是输出的 RAW 图
input, char* output, int width, int	像文件名,width 是输入图像的宽,
height)	height 是输入图像的高。支持 RAW 图
noight)	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	AC IIIIA CELIA INTERIOR INTERI
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
COLS, int M, int sequence[9])	的行, 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
	高。
<pre>void RawToBmp(char* input, char*</pre>	RAW 图像转为 BMP 图像, input 是输
output, int imageWidth, int	入文件名, output 是输出文件名。支
imageHigth)	持宽和高相等的图像。
<pre>void RGBtoHSI(char* input, char*</pre>	
	RGB 模型转为 HIS 模型,input 是输
output)	入文件名, output 是输出文件名。支
output)	入文件名, output 是输出文件名。支持 24 位 BMP 图像。
output) void	入文件名, output 是输出文件名。支持 24 位 BMP 图像。 拉普拉斯锐化滤波器, input 是输入
output)	入文件名, output 是输出文件名。支持 24 位 BMP 图像。

COLS, int M, float w, float mask[3][3])	小,M和w是滤波相关参数,如M=1,w=1; mask 是滤波器模板。支持 RAW 图像。 参考模板: float mask[3][3] = {{0,1,0}, {1,-4,1}, {0,1,0}};
<pre>void RawLaplacianEnhancement(char* input1, char* output1, int width, int height)</pre>	拉普拉斯算子增强,input1 是输入的RAW 图像文件名,output1 是输出的RAW 图像文件名,width 是输入图像的宽,height 是输入图像的高。支持RAW 图像。
<pre>void CyanGray(char* input, char* output, int width, int height)</pre>	青色灰度图像。
void MagentaGray(char* input, char* output, int width, int height)	品红灰度图像。
<pre>void YellowGray(char* input, char* output, int width, int height)</pre>	黄色灰度图像。
void Transfer(char* input, char* output, int width, int height)	传递函数。
<pre>void Homography(char* input1, char* input2, char* input3, char* output, int width, int height, int newwidth, int newheight) void MovieEffect(char* input, char*</pre>	单应。 电影效果。
<pre>output, int width, int height) void FixedThresholdMethod(char* input, char* output, int width, int height)</pre>	抖色处理,固定阈值法。
<pre>void RandomThresholdMethod(char* input, char* output, int width, int height)</pre>	抖色处理,随机阈值法。
<pre>void DitherMatrixMethod(char* input, char* output, int width, int height, int N)</pre>	抖色处理,抖动矩阵法,默认 N=2。
<pre>void NormalizedLogBuffer1(char* input, char* output, int width, int height)</pre>	对数变换,规范化对数。
<pre>void NormalizedLogBuffer2(char* input, char* output, int width, int height)</pre>	对数变换,规范化对数。
<pre>void TernaryGrayLevel1(char* input, char* output, int width, int height)</pre>	三值灰度。

void TernaryGrayLeve12(char*	二估左府
void TernaryGrayLevel2(char* input, char* output, int width, int	三值灰度。
height)	
void BestEdgeMap1(char*	最佳边贴图。
input, char* output, int width, int	取任之如母。
_ · · · .	
height) void BestEdgeMap2(char*	
0 1 \	最佳边贴图。
<pre>input, char* output, int width, int height)</pre>	
void Skeletonize(char* input, char*	骨架化。
output, int width, int height)	H A Pu
void SeparableDiffusion(char*	可分离扩散。
input, char* output, int width, int	
1 - · · · · · · · · · · · · · · · · · ·	
height)	   去除噪声。
void Denoising (char* input1, char*	
<pre>input2, char* output, int width, int height)</pre>	
void Luminosity(char* input, char*	   亮度调整。
output, int width, int height)	九汉 啊 正。
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)	取行一致人。
void Shrink(char* input, char*	收缩。
output, int width, int height)	TAPIN O
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.
void Dewarped2(char* input, char*	脱蜡。a是在输出图像中检查半径是
output, int width, int height, int	否<=a, 然后再进行扭曲, 参考:
Offset, double a, double b, double	Offset=256, a=256.5, b=0.5.
coeffx[12], double coeffy[12])	脱蜡规范:
2 3, 1	double coeffx[12] =
	{ 1.00056776e+00, -5.68880703e-
	04, -1.13998357e-03,
	1. 00056888e+00, -
	5. 65549579e-04, -1. 13554790e-03,
	9. 99434446e-01 ,
L	,

	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};
<pre>void TextureSegmentation1(char* input, char* output, int width, int height, int K, int N)</pre>	纹理分割,默认 K=6, N=100。
<pre>void TextureSegmentation2(char* input, char* output, int width, int height, int K, int N)</pre>	纹理分割,默认 K=6, N=100。
<pre>void TextureClassification(vector</pre>	纹理分类, a 是要分类的图像的数量, 如 filename 里有 3 个图像名称,则 a=3; output 是分类结果文件,格式 为 txt 的文本文件;默认 K=4,N=1000。
void ErrorDiffusion1(char* input, char* output, int width, int height)	误差扩散。
<pre>void ErrorDiffusion2(char* input, char* output, int width, int height)</pre>	误差扩散。
void ErrorDiffusion3(char* input, char* output, int width, int height)	误差扩散。
<pre>void Thin(char* input, char* output, int width, int height) void OilPainting(char* input, char*</pre>	图像细化。 油画效果,默认 N=2。
output, int width, int height, int N) void OilPaintingl(char* input, char* output, int width, int	油画效果,默认 N=2。
height, int N)  void AverageFiltering (char* inputfile, char* outputfile, int	3*3 平均值滤波。

width, int height)	
<pre>void GeometricMeanFiltering(char*</pre>	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)	
void GenerateImage(char*	生成 24 位和 32 位 BMP 图像。type 等于图

```
像的位数,如 type=24。
output, BMPMat**
color, unsigned short type)
                           参考用例:
                           BMPMat**
                                            color
                           (BMPMat**) malloc(sizeof(BMPMat*)*1280
                           ):
                              for (unsigned int i = 0; i < 1280;
                           i++)
                                 color[i]
                           (BMPMat*) malloc(sizeof(BMPMat)*2450);
                              for (unsigned int i = 0; i < 1280;
                           i++) {
                                  for (unsigned int j = 0; j < 0
                           2450; j++) {
                                      color[i][j].B = 0;
                                      color[i][j].G = 0;
                                      color[i][j].R =255;
                                  }
                           直方图均衡,支持8位和16位BMP。input
void
HistogramEqualization5(char*
                           是输入文件名,output 是输出文件名。
input, char* output)
              Resize(char*
                           图片缩放,支持8位和16位BMP。input是
void
                           输入文件名, output 是输出文件名。Height
input, char*
                output, int
                           和 Width 是输出图像的高和宽。
Height, int Width)
                           求图像的平均亮度,支持8位和16位BMP。
doub1e
      MeanBrightness(char*
                           input 是输入文件名。
input)
int IsBitMap(FILE *fp)
                           判断是否是位图。
                           获得图片的宽度。
int getWidth(FILE *fp)
int getHeight(FILE *fp)
                           获得图片的高度。
                           获得每个像素的位数。
unsigned short getBit(FILE
*fp)
                           获得数据的起始位置。
unsigned int getOffSet(FILE
*fp)
                           BMP 图像转为 YUV 图像, input 是输入文件
void
             BMPtoYUV(char*
                           名, output 是输出文件名。yuvmode 是 YUV
input, char*
            output,
                      char
                           文件的 3 个模式选项, yuvmode 的值可为
vuvmode)
                           '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
                           件名, output 是输出文件名。
input, char* output)
                           BMP 图像转为 YUV420 图像, input 是输入文
void
        BMPtoYUV420II (char*
                           件名, output 是输出文件名。
input, char* output)
                           DCM 图像转 BMP 图像。input 是输入文件名,
void
            DCMtoBMP(string
```

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)	IV N. II. V
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)	
	矮小化特效。a=1, b=128, c=0.5, d=0.5,
input, char* output, int a, int	depth=24。文狩24位BMP图像。
b, double c, double d, int	
depth)	
void Handstand(char*	倒立特效。a=1, b=128, c=0.5, depth=24。
input, char* output, int a, int	支持 24 位 BMP 图像。
b, double c, int depth)	田中中分 0-1 L-190 0-0 5 J-141-04
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)	高脚特效。a=1,b=128,c=2,d=0.5,
void HighFoot (char*	
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	achen 710 ×11 74 下 DMI 民隊。
depth)	
void Thin(char* input, char*	细化特效。a=1, b=128, c=0.5, d=0.5,
void inin (chair input, chair	-щ ги ту лдо, с-0, 0, u-0, 0,

output, int a, int b, double	depth=24。支持 24 位 BMP 图像。
c, double d, int depth)	40 bon 510 ×14 51 压 Dim 国家。
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	e-0.5, deptn-24。 文书 24 位 DMI 图像。
d, double e, int depth)	上字注即於河之梅丰上
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;
	) Bill indedouble,
	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 CrossDenoising(BMPMat**	十字法剔除孤立像素点。
input, BMPMat** output, double	需引入以下结构体和声明:
a)	typedef struct {
	unsigned char B; //24 位和 32 位 BMP
	图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP
	图像的绿色通道分量
	unsigned char R; //24 位和 32 位 BMP
	图像的红色通道分量
	unsigned char A; //仅限 32位 BMP 图

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

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

	double B;
	double G;
	double R;
	double A;
	BMPMatdouble;
	) DMF Matdouble,
	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);
woid  Matrix Danaiain (DMDMatrix)	短阵法剔除孤立像素点。 電引入以下结构体和声明
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 图像的蓝色通道公量
	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;
	John Matdouble,
	void Conversion8(unsigned char**
	input, double** output);
	void Conversion8(double**
	input, unsigned char** output);
	void Conversion24(BMPMat**
	input, BMPMatdouble** output);
	void Conversion24(BMPMatdouble**
	, ·
void ImageFusion3(char*	input,BMPMat** output); 多聚焦图像的融合,支持 8 位 BMP 图像。
,	多乘馬图像的融音,又持 o 位 DMF 图像。 block height=8 , block width=8 ,
input1, char* input2, char*	niock_Height-o , block_width-o ,

output, int block_height, int	threshold=1.75。
block_width, double	
threshold)	
void ImageFusion4(char*	多聚焦图像的融合,支持 8 位 BMP 图像。
input1, char* input2, char*	block_height=8 , block_width=8 ,
output, int block_height, int	threshold=1.75.
block_width, double	
threshold)	
void ImageFusion5(char*	图像融合。参考: a=3, b1=4, DX1=-68, DY1=-
input1, char* input2, char*	99, EPS=1, input1="图像融合 1.jpg",
MaskImage, char* output, int	input2="图像融合 2.jpg", MaskImage="掩
dx[], int $dy[]$ , int a, double	膜.png", output="output.jpg"。
b1, int DX1, int DY1, double	int $dx[] = \{0, 0, -1, 1\};$
EPS)	int $dy[] = \{-1, 1, 0, 0\};$
void Screenshot3(HWND hWnd,	截屏函数。hWnd 是要截屏的窗口句柄,如:
LPCWSTR OutputImage)	GetDesktopWindow();OutputImage 截图名
	称。
void Screenshot1 (HWND hWnd,	截屏函数。hWnd 是要截屏的窗口句柄,如:
LPCWSTR OutputImage)	GetDesktopWindow();OutputImage 截图名
· · · · · · · · · · · · · · · · · · ·	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
il Daul-(al-aut-in-aut-al-aut-	称。
void Dark(char* input, char*	暗调滤镜,参考: ratio=100。
output, int ratio) void WaveFilter(char*	冰泊亚亦株为海傍 10,77000 具海侩积度 (冰
input, char* output, int	波浪形变特效滤镜, degree 是滤镜程度(波 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
degree, int a)	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,
input, chai. output, int a/	a=3 时生成 TGA 图像。
void PinchFilter(char*	挤压形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
,,	坐标, cenY 是形变中心点 Y 坐标。
void SpherizeFilter(char*	球面形变特效滤镜,a=0时生成BMP图像,
input, char* output, int a)	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
	a=3 时生成 TGA 图像。
void SpherizeFilter(char*	球面形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像,cenX 是形变中心点 X
	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,

	A MILL D. TRO EDIZ.
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。
<pre>void ClosedOperation(char*</pre>	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(	
string input, char* output)	
CImg <unsigned int=""></unsigned>	HSI 空间。
HistEqualColorImageHSISpace(	
	1

string input)	or h hlith
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。
char *output, unsigned int	
w, unsigned int h)	
void	彩色图像垂直镜像, input 是输入图像的像
ColorImageVerticalMirror(uns	素数据, output 是输出图像的像素数据, w
igned char *input, unsigned	是输入图像的宽,h 是输入图像的高。
char *output, unsigned int	
w, unsigned int h)	
void OTSU(char* input, char*	大津算法, input 是输入文件名, output 是
output, int BeforeThreshold)	输出文件名。BeforeThreshold 是初始阈
	值,如 BeforeThreshold=10。支持 8 位 BMP
	图像。
void LowerBrightness(char*	调低亮度, input 是输入文件名, output 是
<u> </u>	

input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b) (1 - H; 1 - P; 1 (1 +	的参考值可为 a=100,b=0。
void HightBrightness(char*	调高亮度, input 是输入文件名, output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100,b=0。
void	迭代阈值选择, input 是输入文件名,
IterativeThresholdSelection(	output 是输出文件名。支持 8 位 BMP 图像。
char* input, char* output)	
void DitheringMethod(char*	抖动法, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void LogTransformation(char*	对数变换, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。constant
constant)	是相关参数,如 constant=15。
void	对数变换, input 是输入文件名, output 是
LogarithmicTransformation(ch	输出文件名。支持 BMP 图像。
ar* input, char* output)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization(char*	output 是输出文件名。支持 BMP 图像。
input, char* output)	
void Binarization(char*	二值化, input 是输入文件名, output 是输
input, char* output, int	出文件名。支持 24 位 BMP 图像。threshold
threshold)	是阈值,如: threshold=128。
void Expansion(char*	二值图像膨胀,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。
void Corrosion(char*	二值图像腐蚀,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。
void OpenOperation(char*	二值图像开运算,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。
void ClosedOperation(char*	二值图像闭运算,参考:
input, char* output, unsigned	mask[9]={0,255,0,255,255,255,0,255,0}
char mask[9], int c)	, c=128。
void	二值图像开运算提取轮廓,参考:
OpenOperationToExtractContou	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
r(char* input, char*	, c=128°
output, unsigned char	
mask[9], int c)	
void	二值图像膨胀运算提取轮廓,参考:
ExpansionOperationToContourE	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
xtraction(char* input, char*	, c=128.
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	, 6 1200
char mask[9], int c)	
void Glaw(char* input, char*	发光滤镜,参考: ratio=100。
output, int ratio)	7, 10010 1000
void LowPassFilter(char*	低通滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 BMP 图像。
void HighPassFilter(char*	高通滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 BMP 图像。
void Thinning(char*	图像细化, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 BMP 图像。
void ThinningLine(char*	图像细化且线条化, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 BMP 图像。
void Corrosion(char*	腐蚀, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 4 位 BMP 图像。
void Corrosion1(char*	腐蚀, input 是输入文件名, output 是输出
input, char* output, int	文件名。支持 24 位 BMP 图像。TempBuf 是
*TempBuf, int TempH, int	腐蚀模板,TempH 和 TempW 分别是 TempBuf
TempW)	的高和宽,如 TempH=4, TempW=4,则有
	TempBuf[4][4].
void Expand (char*	膨胀, input 是输入文件名, output 是输出
input, char* output, int	文件名。支持 24 位 BMP 图像。TempBuf 是
*TempBuf, int TempH, int	膨胀模板,TempH 和 TempW 分别是 TempBuf
TempW)	的高和宽,如 TempH=4, TempW=4,则有
	TempBuf[4][4].
unsigned char**	线性存储的灰阶图像像素转化为二维。
create2DImg(unsigned char*	
input, int w, int h)	
unsigned char	图像指定区域取最大值(判断是否超出边
getMaxPixelWhole(unsigned	界)。
char **input, int x, int y, int	
w, int h, int *Kernal, int	
kernalW, int halfKernalW)	网络化宁区械而县十估/不如此目不知山丛
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	
rneinai, int Keinaiw, 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	7170
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	1 124 / PA
*input, int w, int h, int	
*Kernal, int kernalW, int	
halfKernalW)	
void Corrosion(unsigned char	二值腐蚀。
*input, unsigned char	
*output, int rows, int	
cols, int mat[5][5])	
void Expansion(unsigned char	二值膨胀。
*input, unsigned char	
*output, int rows, int	
cols, int mat[5][5])	
void	高斯滤波,支持 PNG 文件。
GaussianBlurFilter(char*	
input, char* output)	
void GaussianFiltering(char*	高斯滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void	拉普拉斯增强, input 是输入文件名,
LaplaceEnhancement(char*	output 是输出文件名。支持 24 位 BMP 图
input, char* output)	像。
void Residual(char*	求残差, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void SunlightFilter(char*	光照特效滤镜, intensity 是光照强度, 如:
input, char* output, int	intensity=255; radius 是光照范围,如:
intensity, int radius, int	radius=600; x 和 y 是光照的位置, 如:
x, int y)	x=100, y=60。
void Compress(char*	压缩,支持多种文件。input 是要压缩的文
input, char* output)	件名,output 是压缩后的文件名。
void Decompression(char*	解压缩,支持多种文件。input 是要解压缩
input, char* output)	的文件名,output 是解压缩后的文件名。
void BlackWhite(char*	黑白化, input 是输入的原图像, output 是
input, char* output)	输出的黑白图像。支持 24 位 BMP 图像。

void Underexposure(char* input, char* output)	图像欠曝光, input 是输入的原图像, output 是输出的欠曝光图像。支持 24 位 BMP 图像。
void Overexposure(char* input, char* output)	图像过曝光, input 是输入的原图像, output 是输出的过曝光图像。支持 24 位 BMP 图像。
void Nostalgia(char* input, char* Mask, char* output, int ratio)	怀旧滤镜, input 和 Mask 都是输入的文件 名, Mask 是褶皱图像路径, ratio=100。
<pre>void GammaTransform(char* input, char* output)</pre>	伽马变换, input 是输入文件名, output 是输出文件名。支持 8 位 BMP 图像。
void GrayScale(char* input, char* output)	灰度化, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。 灰度图二值化, bit 用于设定位数, 如
GrayImageBinarization(char* input, char* output, int bit, int threshold)	bit=8; threshold 是 阈 值 , 如 threshold=200。支持 8 位 BMP 图像。
<pre>void GreyPesudoColor(char* input, char* output)</pre>	灰度图伪彩色化, input 是输入文件名, output 是输出文件名。支持 8 位 BMP 图像。
<pre>void CalculateCumulativeHistogram Map(char* input, char* outfile)</pre>	计算累加直方图并映射, input 是输入文件名, outfile 是输出文件名。支持24位BMP图像。
void Translation(string input, char* output, int dx, int dy)	图像平移, input 是输入的文件, dx 和 dy 是横向及纵向的移动距离 (像素), 负值是 向左 / 向下移动; output 是平移操作后的 结果文件名。支持 BMP 图像。
<pre>void</pre>	镜像变换, input 是输入的文件, output 是 镜像操作后的结果文件名, axis 是镜像变 换的方向(以 X 或 Y 表示)。支持 BMP 图像。
void Sheared(string input, char* output, char axis, double Coef)	错切变换, input 是输入的文件, output 是错切操作后的结果文件名, axis 和 Coef 分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 input, char* output, double cx, double cy)	缩放操作, input 是输入的文件, output 是缩放操作后的结果文件名, cx 和 cy 分别是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。
<pre>void Rotated1(string input, char* output, double angle)</pre>	图像旋转, input 是输入的文件, output 是图像旋转后的结果文件名, angle 是旋转角度, 弧度制。支持 BMP 图像。
void SaltNoise(char* input, char* output, int a, int	添加椒盐噪声, a 和 b 是噪声相关参数, 如 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*	(d-255。 文持 6 位 DMF 图像。 交叉冲印滤镜,参考: ratio=100。
	文文件中愿说,参与: Tat10-100。
input, char* output, int	
ratio)	· 1 1
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)	L. L
void Conversion8(unsigned	unsigned char**转 unsigned int**,
char** input, unsigned int**	output 用于保存结果(与 input 大小相同)。
output)	
void Conversion8(unsigned	unsigned int**转 unsigned char**,
int** input, unsigned char**	output 用于保存结果(与 input 大小相同)。
output)	
void Conversion8(unsigned	unsigned char**转 float **, output 用于
char** input, float** output)	保存结果(与 input 大小相同)。
void Conversion8(float**	float **转 unsigned char**, output 用于
input, unsigned char**	保存结果(与 input 大小相同)。
output)	
void Conversion8(unsigned	unsigned char**转double **, output用
char** input, double**	于保存结果(与 input 大小相同)。
output)	
void Conversion8(double**	double **转unsigned char**,output用
input, unsigned char**	于保存结果(与 input 大小相同)。
output)	
void Conversion8(unsigned	unsigned char**转 char **, output 用于
char** input, char** output)	保存结果(与 input 大小相同)。
void Conversion8(char**	char **转 unsigned char**, output 用于
input, unsigned char**	保存结果(与 input 大小相同)。
output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatshort **, output 用于
input, BMPMatshort** output)	保存结果(与 input 大小相同)。
void	BMPMatshort **转 BMPMat **, output 用于
Conversion24(BMPMatshort**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatint **, output 用于保
input, BMPMatint** output)	存结果(与 input 大小相同)。
void	BMPMatint **转 BMPMat **, output 用于保
L .	* /:- * */*

0.4/pupu	<b>左</b> 休田 / 上:
Conversion24 (BMPMatint**	存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion24(BMPMat**	BMPMat **转 BMPMatfloat **, output 用于
input,BMPMatfloat** output)	保存结果(与 input 大小相同)。
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)	
void Conversion32(BMPMat**	BMPMat **转 BMPMatint **, output 用于保
input, BMPMatint** output)	存结果(与 input 大小相同)。
void	BMPMatint **转 BMPMat **, output 用于保
Conversion32(BMPMatint**	存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion32(BMPMat**	BMPMat **转 BMPMatfloat **, output 用于
input, BMPMatfloat** output)	保存结果(与 input 大小相同)。
void	BMPMatfloat **转 BMPMat **, output 用于
Conversion32(BMPMatfloat**	保存结果(与 input 大小相同)。
input, BMPMat** output)	
void Conversion32(BMPMat**	BMPMat **转BMPMatdouble **, output用
input, BMPMatdouble** output)	于保存结果 (与 input 大小相同)。
void	BMPMatdouble **转 BMPMat **, output 用
Conversion32(BMPMatdouble**	于保存结果 (与 input 大小相同)。
input,BMPMat** output)	
void Conversion32(BMPMat**	BMPMat **转 BMPMatchar **, output 用于
input, BMPMatchar** output)	保存结果(与 input 大小相同)。
void	BMPMatchar **转 BMPMat **, output 用于
Conversion32(BMPMatchar**	保存结果(与 input 大小相同)。
input,BMPMat** output)	
void MeanFiltering(char*	均值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。

void MeanFltering1(char*	均值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
void KapoorAlgorithm(char*	卡普尔算法,input 是输入文件名,output
input, char* output, int	是输出文件名。BeforeThreshold 是初始阈
BeforeThreshold)	值,如 BeforeThreshold=150。支持 8 位 BMP
	图像。
void OpenOperation(char*	开运算, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 4 位 BMP 图像。
void Diffusion(char*	扩散滤镜,参考: ratio=90。
input, char* output, int	J. 14010 000
ratio)	
void LapulasFiltering(char*	拉普拉斯滤波, readPath 是原图像,
readPath, char*	writePath 是处理后的图像文件名。支持 8
writePath, float	位 BMP 图像。
CoefArray[9], float coef)	各参数参考值:
	定义*3 的模板 (拉普拉斯):
	float
	CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0
	f, 2. 0f, 1. 0f, 2. 0f, 1. 0f};
	定义模板前乘的系数(拉普拉斯):
	float coef=(float) (1.0/16.0);
void ImageFiltering(char*	图像滤波, input 是输入文件名, output 是
input, char* output, float	输出文件名。kernel 是模糊内核。支持 24
kerne1[3][3])	位 BMP 图像。
void ComicStrip(char*	连环画滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void	亮度对比度调节,参考: brightness=-30,
BrightnessAdjustment1(char*	contrast=100.
input, char* output, int	
brightness, int contrast)	
void	亮度对比度调节,参考: brightness=-30,
BrightnessAdjustment2(char*	元及为记及调节,参考: Dirighthess—50, contrast=100。
input, char* output, int	
brightness, int contrast)	最持大上升投护员 土柱 0 比和 04 比 pup
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)	<b>アンサンロ上位                                    </b>
void PopArtStyle(char*	流行艺术风滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void LightLeakage(char*	漏光滤镜, input 和 Mask 都是输入的图像
input, char* Mask, char*	名,Mask 是漏光模板图像,ratio=90。
output, int ratio)	
output, int ratio)	

void LinearFiltering(char*	线性滤波, input 是输入文件名, output 是
<pre>input, char* output, short average[3][3])</pre>	输出文件名。支持 8 位 BMP 图像。 参考模板:
average[3][3])	多名模拟:   short average[3][3] = {{1, 2, 1},
	{2, 4, 2},
	{1, 2, 1}};
void MedianFiltering(char*	中值滤波, input 是输入文件名, output 是
input, char* output, short average[3][3])	输出文件名。支持 8 位 BMP 图像。 参考模板:
	short average[3][3] = $\{\{1, 2, 1\},$
	$\{2, 4, 2\},$
void	{1, 2, 1}}; 锐化滤波, input 是输入文件名, output 是
SharpeningFiltering(char*	输出文件名。支持8位BMP图像。
input, char* output, short	参考模板:
average[3][3], short	short average[3][3] = $\{\{1, 2, 1\},$
sharpen[3][3])	$\{2, 4, 2\},$
	$\{1, 2, 1\}\};$ short sharpen[3][3] = $\{\{-1, -1, -1\},$
	$\{-1, 8, -1\},\$
	$\{-1, -1, -1\}\};$
void	梯度锐化, input 是输入文件名, output 是
GradientSharpening(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output, short average[3][3], short	参考模板:   short average[3][3] = {{1, 2, 1},
soble1[3][3], short	Short average[3][3] = $\{\{1, 2, 1\}, \{2, 4, 2\}, \}$
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 GeometricMeanFilter(char*	几何平均滤波器, input 是输入文件名, output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	Output 定制山入川石。又行 O JU DMI 国家。
void	调和平均滤波器, input 是输入文件名,
HarmonicMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	反调和平均滤波器, input 是输入文件名,
ContraHarmonicMeanFilter(cha	output 是输出文件名。支持 8 位 BMP 图像。

r* input, char* output)	
void Filter(char*	滤波 input 具绘 λ 立併夕 output 且绘山
input, char* output)	滤波, input 是输入文件名, output 是输出 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
	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。
void MidSmoothing(char*	中值滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void AvgSmoothing(char*	均值滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void Averaging(char*	图像平均化, input 是输入文件名, output
input1, char* input2, char*	是输出文件名。a 是平均化相关参数,如
input3, char* output, int a)	a=3。支持 8 位 BMP 图像。
void PlaneSlicing(char*	平面切片, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void Translation(char*	图像平移,参考: xoffset=-100, yoffset=-
input, char* output, int	100。
xoffset, int yoffset)	
void	锐化空间滤波器, input 是输入文件名,
SharpeningSpatialFiltering8(	output 是输出文件名。model 是锐化模板。
char* input, char* output, int	支持8位灰度图像。
mode1[9])	
void PseudoGrayscale(char*	
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 是要填充的两个颜色。支持 24 位 BMP 图像。
color1, unsigned char color2) void	元的两个颜色。文持 24 位 BMP 图像。 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)	W WORT ET IN 0
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] =
	$\{0.0, 0.0, 1.0\}, S[3] = \{0.0, 0.0, 0.0\}$
	1.0}, r=50, a=0.5, m=1。支持 RAS 文件。
	) == == > <del></del>

<pre>void    MakeSphere(double vector_v[3], double vector_s[3], double r, double a, double m, int ROWS, int COLS, char* output, double max)  void BilateralFiltering(string input, char* output, double ssd, double sdid)</pre>	使用反射模型生成球体的图像,vector_v 是摄影机的方向,output 是输出的结果图 像文件名,ROWS 是输出图像的行数,COLS 是输出图像的列数,参考: vector_v[3] = {0.0, 0.0, 1.0}, vector_s[3] = {0.0, 0.0, 1.0}, r=50, a=0.5, m=1。支持 RAS 文件。 双边滤波,input 是输入文件名,output 是 输出文件名。支持 24 位 BMP 图像。ssd 和 sdid 分别是空间域标准差与强度域标准 差。
void DoubleLayerErosion(char* input, char* output)	具有圆形结构集的双层形态侵蚀,支持 8 位和 24 位 BMP 图像。
void BinaryImageHorizontalMirror( unsigned char *input, unsigned char *output, unsigned int w, unsigned int h)	二值图像水平镜像。
void GrayImageHorizontalMirror(un signed char *input, unsigned char *output, unsigned int w, unsigned int h)	灰阶图像水平镜像。
void ColorImageHorizontalMirror(u nsigned char *input, unsigned char *output, unsigned int w, unsigned int h)	彩色图像水平镜像。
<pre>void SketchFilter(char* input, char* output, int ratio)</pre>	素描滤镜,参考: ratio=100。
<pre>void Zoom(char* input, char* output, float scaleX, float scaleY, int interpolation)</pre>	缩放,参考: scaleX=5, scaleY=5, interpolation=0或interpolation=1。
void AddGaussNoise(char* input, char* output)  void AddSaltPepperNoise(char* input char* output)	添加高斯噪声, input 是输入文件名, output 是输出文件名。支持8位BMP图像。添加椒盐噪声, input 是输入文件名, output 是输出文件名。支持8位BMP图像。
<pre>input, char* output) void ChannelSeparation(char* input, char* Routput, char* Goutput, char* Boutput)</pre>	通道分离, input 是输入文件名, Routput 是红色通道图像, Goutput 是绿色通道图 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)	持 24 位 BMP 图像。

图像无损解压,input 是待解压的文件名,output 是输出解压后的 BMP 文件名。支持 24 位 BMP 图像。 void
har* input, char* output)  void  ImageDiscoloration(char* input, char* output, double a, double b, double c)  unsigned Char** HorizontalConcavity(unsigned char** input, int RANGE, int  24 位 BMP 图像。  图像变色, input 是输入文件名, output 是输出文件名。如: a=0.2126, b=0.7152, c=0.0722。支持 24 位 BMP 图像。  图像变形之水平内凹,返回处理结果。参考: RANGE=400。
void ImageDiscoloration(char* input, char* output, double a, double b, double c) unsigned char** HorizontalConcavity(unsigned char** input, int RANGE, int
ImageDiscoloration(char* input, char* output, double a, double b, double c)  unsigned char** HorizontalConcavity(unsigned char** input, int RANGE, int RA
input, char* output, double c=0.0722。支持 24 位 BMP 图像。 a, double b, double c)  unsigned char** HorizontalConcavity(unsigned char** input, int RANGE, int
a, double b, double c) unsigned char** HorizontalConcavity(unsigned char** input, int RANGE, int
unsigned char** 图像变形之水平内凹,返回处理结果。参考: HorizontalConcavity(unsigned char** input, int RANGE, int
HorizontalConcavity(unsigned char** input, int RANGE, int
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。
char *src, int w, int h, float [in] src: 图像, 单通道
scaleX, float scaleY, [in] w: 宽
boundingbox_t bbox, [in] h: 高
std::vector <line_float_t> [in] scaleX: X 轴上的缩小因子</line_float_t>
&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
return: 0:ok; 1:error
需引入以下结构体:
typedef struct
{
int x;
int y;
int width;
int height;
} boundingbox_t;

```
typedef struct
                              float startx;
                              float starty;
                              float endx;
                              float endy;
                           }line float t;
int
                           边缘划线检测器。
                                              图像,单通道
EdgeDrawingLineDetector (unsi
                           [in]
                                    src:
gned char *src, int w, int
                           [in]
                                                   宽
                                     w:
h, float scaleX, float scaleY,
                           [in]
                                     h:
                                                   高
                                              X轴上的缩小因子
                           [in]
                                  scaleX:
boundingbox t
                     bbox,
std::vector<line float t>
                           [in]
                                  scaleY:
                                              Y轴上的缩小因子
                                              要检测的边界框
&lines)
                           [in]
                                   bbox:
                                               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;
                           传播滤波器。
int
                                                  输入图像
PropagatedFilter1 (unsigned
                           [in]
                                    src:
                                                  引导图像
char
     *src,
            unsigned
                      char
                           [in]
                                     guidance:
           unsigned
                           [in/out]]
                                      dst:
                                                  输出图像
*guidance,
                      char
                           [in]
*dst, int w, int h, int c, int
                                                   宽
                                    w:
                                                   高
r, float
           sigma s,
                     float
                           [in]
                                    h:
                                        图像通道,仅 c=1 或 c=3
sigma r)
                           [in]
                                    c:
                           [in]
                                            局部窗口半径
                                    r:
                           [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
```

	D.D. best
	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。
	<pre>@return: 0:ok; 1:error</pre>
int	传播滤波器。
PropagatedFilter2(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 BoxfilterFilter(unsigned	
	[in] src: 输入图像,单通道
*dst, int w, int h, int c, int	
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

structure and the structure of	[:/] 1-4. 松山圆角 苗涌送
*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:向上缩放< td=""></sr<1:向上缩放<>
	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]
	[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	霍夫线探测器。
HoughLineDetector(unsigned	[in] src: 图像,单通道
char *src, int w, int h, float	[in] w: 宽
scaleX, float scaleY, float	[in] h: 高
CannyLowThresh, float	[in] scaleX: X 轴上的缩小因子
CannyHighThresh, float	[in] scaleY: Y轴上的缩小因子
HoughRho, float HoughTheta,	[in] CannyLowThresh: canny 算子中迟滞
<u> </u>	V V V V V V V V V V V V V V V V V V

float MinThetaLinelength, float MaxThetaGap, int HoughThresh, HOUGH\_LINE\_TYPE\_ CODE \_type, boundingbox\_t bbox,

std::vector<line\_float\_t>
&lines)

过程的低阈值

[in] CannyHighThresh: canny 算子中迟滞过程的高阈值

[in] HoughRho: 累加器的距离分辨率(以像素为单位)

[in] HoughTheta: 累加器的角度分辨率 (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) [in] \_type: hough 线 方 法: hough\_line\_STANDARD 或 hough\_line PROBABILISTIC

[in] bbox: 要检测的边界框

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

unsigned char \*dst, int w, [in] 宽 w:int h, int c, float sigma s, [in] h: 高 float sigma r, float scale) [in] 图像通道,仅 c=1 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\*255return: 0:ok; 1:error void 快速双边滤波器 RGB 通道。 src: 输入图像, RGB 通道 fast bilateral filter color [in] dst:输出图像,RGB通道 (unsigned char \*src, unsigned [in/out] int w, int h, [in] 宽 char \*dst, w: [in] h: 高 float sigma s, float sigma r, float scale) [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\*255return: 0:ok; 1:error 快速双边滤波器 RGB 通道。 int src: 输入图像, RGB 通道 fast bilateral filter color( [in] dst: 输出图像, RGB 通道 unsigned char \*src, unsigned [in/out] char \*dst, int w, int h, int [in] 宽 w:float sigma s, float [in] h: 高 图像通道,仅 c=3 sigma\_r, float \_scale) [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\*2550:ok; 1:error return: 快速双边滤波器。 FastBilateralFilter (unsigned [in] 输入图像 src: 引导图像,单个通 char \*src, unsigned [in] guidance: char 道, 只有单个通道有效 \*guidance, unsigned char \*dst, int w, int h, int c, [in/out] dst: 输出图像 float sigma s, float [in] w: 宽 高 sigma r, float scale) [in] h: [in] c: 图像通道,仅 c=1 或 c=3 sigma s: 坐标空间中的滤波 [in] 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 [in] 西格玛。该参数的值越大, 意味着像素邻域 (请参见 sigmaSpace) 内更远的颜色将混 合在一起,从而产生更大的半等色区域。 如果正则化, scale = 1; 如果不正则化,  $_{\text{scale}} = 255*255$ return: 0:ok; 1:error 如果引导为 NULL, 仍然可以获得滤色器 快速双边滤波器。 int 输入图像 permutohedral bilateral filt |in| src: 引导图像 er (unsigned char [in] guidance: \*src, 输出图像 unsigned char \*guidance, [in/out] dst: 宽 unsigned char \*dst, int w, int [in] w:h, int c, float sigma s, [in] 高 h: float sigma r, float scale) 图像通道,仅 c=1 或 [in] c:c=3[in] sigma s: 坐标空间中的滤波器 西格玛。参数的值越大,意味着只要颜色足 够接近,更远的像素就会相互影响(请参见 sigmaColor)。当 d>0 时,它指定邻域大小, 而不考虑 sigmaSpace。否则, d 与

	sigmaSpace 成比例。
	[in] sigma_r: 颜色空间中的滤波器
	西格玛。该参数的值越大, 意味着像素邻域
	(请参见 sigmaSpace)内更远的颜色将混
	合在一起,从而产生更大的半等色区域。
	如果正则化, _scale = 1; 如果不正则化,
	scale = 255*255
	return: 0:ok; 1:error
	try: (src, guidance, dst, w, h, c, 1.6f, 0.6f
	, 255*255)
void HighPassFilter(char*	高通滤波器,参考: preserve=0。
input, char* output, int	同心心权福,多写:preserve o。
preserve)	
void EmbossFilter(char*	   浮雕过滤器,参考: preserve=1。
· ·	1丁州E人工协心有的, 多/与: pieseive-i。
input, char* output, int	
preserve)	份
void SharpenFilter(char*	锐化过滤器,参考: preserve=1。
input, char* output, int	
preserve)	<b>业</b> 10
void Convolution(char*	卷积,参考: w=7, preserve=1。
input, char* output, int w, int	
preserve)	
void GaussianBlur(char*	高斯模糊,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	
void HybridImage(char*	混合图像,参考: sigma=2, preserve=1。
input1, char* input2, char*	
output, float sigma, int	
preserve)	
void LowFrequencyImage(char*	低频图像,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage(char*	
input, char* output, float	
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImagel(char*	_
input, char* output, float	
sigma, int preserve)	
void Bilateral(char*	双边滤波,参考: sigmal=3, sigma2=0.1。
input, char* output, float	_
sigmal, float sigma2)	
void SkinSmooth(char*	皮肤细滑, a 是平滑级别, b 代表是否应用
	, , , , , , , , , , , , , , , , , , , ,

input, char* output, int a, int	皮肤过滤器,a=2,b=1。
b)	
void Resize1(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)	
void EqualizedGray(char*	灰度图直方图均衡化。
input, char* output)	
void	彩色图直方图均衡化。
ColorHistogramEqualization(c	
har* input, char* output)	
void AverageHistogram(char*	直方图均衡化。
input, char* output)	
void HSIHist(char* input,	HIS 直方图。
char* output)	
void ImageCutting(char*	图像裁剪, input 是输入文件名, output 是
input, char* output, int	输出文件名。leftdownx,leftdowny,
leftdownx, int leftdowny, int	rightupx,rightupy 是要裁剪的矩形区域
rightupx, int rightupy)	的左下角和右上角的坐标(连续四个整数

	值,如 50 50 300 300)。支持 24 位 BMP 图像。
void	图像层算法。
ImageLayerAlgorithm(char*	H M/A TIA
input, char* output)	
void	图像无 LUT 的灰度化, input 是输入文件
RGBtoGraywithoutLUT(char*	名,output 是输出文件名。支持 24 位 BMP
input, char* output)	图像。
void RGBtoGraywithLUT(char*	图像有 LUT 的灰度化, input 是输入文件
input, char* output)	名,output 是输出文件名。支持 24 位 BMP
input, chair output,	图像。
void	分段线性变换, input 是输入文件名,
PiecewiseLinearTransform(cha	output 是输出文件名。支持 8 位 BMP 图像。
r* input, char* output)	
void PowerConvertion(char*	功率转换, input 是输入文件名, output 是
input, char* output, double	输出文件名。如: c = 1.2, g = 0.5。支持
c, double g)	8位BMP图像。
void Smooth(char*	平滑, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 8 位 BMP 图像。
void 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:
LaplMask[3][3])	int LaplMask[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
	-1, 0, 1 };
void Add(char* input, char*	图像添加, input 是输入文件名, output 是
output, int N, int	输出文件名。如:N=1。支持8位BMP图像。
SblMask1[3][3], int	一十二。 文持 6 位 DMF 图像。   参考模板:
Sb1Mask2[3][3], int	多名英似:   int LaplMask[3][3] = {
LaplMask[3][3])	0, 1, 0,
Edimon[0][0])	1, -4, 1,
	1, 1, 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 Lap1Mask[3][3] = {
Sb1Mask2[3][3], int
                                              0, 1, 0,
Lap1Mask[3][3])
                                                1, -4, 1,
                                               0, 1, 0
                                  };
                                  int Sb1Mask1[3][3] = {
                                              -1, -2, -1,
                                               0, 0, 0,
                                                1, 2, 1
                                  };
                                  int Sb1Mask2[3][3] = {
                                              -1, 0, 1,
                                              -2, 0, 2,
                                              -1, 0, 1
                              黑白化图像, input 是输入文件名, output
void
            BlackWhite(char*
input, char* output)
                              是输出文件名。支持 24 位 BMP 图像。
       RandomOperation(char*
                              随意操作, input 是输入文件名, output 是
void
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
                  color2, int
color1, int
color3, int
                  color4, int
```

1.5	
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	高斯平滑, input 是输入文件名, output 是
GaussSmoothSharpen(char*	输出文件名。Template 是高斯平滑模板,
input, char* output, int	coefficient=16。支持 24 位 BMP 图像。
Template[3][3], int	
coefficient)	
void SmoothSharpen(char*	平滑, input 是输入文件名, output 是输出
input, char* output, int	文件名。Template 是平滑模板,均一化处
Template[3][3], int	理, coefficient1 = 9。支持24位BMP图
coefficient)	像。
static void	参考: a=0.33, b=0.33, c=0.33。支持24位
NonmaximumWithoutDoubleThres	BMP 图像。
holding(const LPCTSTR input,	
const LPCTSTR output, double	
a, double b, double c)	
void AdjustPixel(char*	调整像素值, input 是输入文件名, output
input, char* output, int a)	是输出文件名。a 是用于设置图像像素的相
	关参数,如 a=3。支持 24 位 BMP 图像。
void	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	1 . I <del> //</del> / !
	直方图均衡化。
HistogramEqualization(short*	直万图均衡化。

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])	
void DFTRead(short** input,	二维离散傅里叶变换,实部图像。
double** output, long	
height, long width)	
void DFTImaginary(short**	二维离散傅里叶变换,虚部图像。
input, double** output, long	
height, long width)	
void FreSpectrum(short	傅里叶变换的平移。
**input, short **output,long	
height, long width)	
void IDFT(double**	二维离散傅里叶反变换。
re_array, double**	
im_array, short** output, long	
height, long width)	
void	添加高斯噪声。
AddGaussianNoise(short**	
input, short** output, long	
height, long width)	
void	添加椒盐噪声。
AddSaltPepperNoise(short**	
input, short** output, long	
height, long width)	
void MeanFilter(short**	均值滤波器。
input, short** output, long	
height, long width)	
void	几何均值滤波器,默认 product=1.0。

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

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

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

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

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

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

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

<u> </u>	
	}
	}
	GenerateImage8(outputfile,output);
void Relief(BMPMat**	浮雕效果,默认 value=128。
input, BMPMat** output, int	
value)	
void Relief(unsigned char**	浮雕效果,默认 value=128。
input, unsigned char**	11 MEDICAL MARKET VALUE 1200
output, int value)	
void Sharpening (BMPMat**	图像锐化,默认 degree=0.3。
	图像优化,纵似 degree-0.5。 
input, BMPMat** output, double	
degree)	<i>国体队</i> 儿 图 1 00
void Sharpening (unsigned	图像锐化,默认 degree=0.3。
char** input, unsigned char**	
output, double degree)	
void Soften(BMPMat**	图像柔化,默认 value=9。
input,BMPMat** output,int	
value)	
void Soften(unsigned char**	图像柔化,默认 value=9。
input, unsigned char**	
output, int value)	
<pre>void flipX(char* input, char*</pre>	X 方向翻转,支持 JPG 文件。
output)	
void flipY(char* input, char*	Y 方向翻转,支持 JPG 文件。
output)	
void Crop(char* input, char*	裁剪。
output, uint16_t start_x,	
uint16_t start_y, uint16_t	
new_height, uint16_t	
new_width)	
void Resize(char*	缩放。
input, char* output, int	
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)	202 1 4 E
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)	
	- クロスト
void PixeLize(char*	像素化,参考: strength=2。

in and the same	
input, char* output, int	
strength)	<u> </u>
void GaussianBlur(char*	高斯模糊,参考: strength=2。
input, char* output, int	
strength)	
void EdgeDetection(char*	边缘检测,参考: cutoff=115。
input, char* output, double	
cutoff)	
void Sharpen(char*	锐化。
input, char* output)	
void CannyProcessing(char*	Canny 处理,a 可以为 1、2、3、4、5。支持
input, char* output, int a)	BMP 图像。
void AverageGrayScale(char*	平均灰度化。
input, char* output)	
void SimpleBW(char*	简易 BW。
input, char* output)	
void AdvancedBW(char*	高级 BW。
input, char* output)	
void UniformNoise(char*	均匀噪声。
input, char* output)	
void GaussianNoise(char*	高斯噪声。
input, char* output, double	
sigma)	
void	椒盐噪声。
SaltAndPepperNoise(char*	
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)	
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)	→ LE YU。
void	分离出蓝色通道。
ChannelSeparation B(char*	力为山血已起地。
input, char* output)	
void	分离出绿色通道。
ChannelSeparation G(char*	刀肉山绿色通道。
<u> </u>	
input, char* output)	分离出红色通道。
ChannelSeparation R(char*	刀肉山红色通道。
input, char* output)	
void Inverse (char*	反转。
input, char* output)	<u> </u>
void	   直方图均衡化。
HistogramEqualization8(char*	旦刀 囟均舆化。 
input, char* output) void Smooth(char*	平滑。
<pre>input, char* output) void</pre>	Canny 算子。
input, char* output)	Callify 昇 1 。
void EdgeEnhance (char*	边缘增强。
input, char* output)	<b>以</b> 练增强。
	innut 目绘)文件名 output 目绘山文件
void AvrFilter(char* input, char* output1, char*	input 是输入文件名,output 是输出文件
output2, int M, int N)	名。如 M=21,N=1。支持 8 位 BMP 图像。
void GryOppositionSSE(char*	input 是输入文件名,output 是输出文件
	名。支持8位BMP图像。
<pre>input, char* output) void MedianFilter(char*</pre>	中值滤波器,input 是输入文件名,output
·	上输出文件名。如 M=5, N=5。支持 8 位 BMP
input, char* output, int M, int N)	是制出文件名。如 M-5,N-5。文持 8 位 BMP │   图像。
void EdgeSharpeningGry(char*	input 是输入文件名, output 是输出文件 名。支持 8 位 BMP 图像。
<pre>input, char* output) void SJGryandRiceTest(char*</pre>	input 是输入文件名,output 是输出文件
input, char* output)	名。支持8位BMP图像。
void TextTest(char*	input 是输入文件名,output 是输出文件
input, char* output)	名。支持8位BMP图像。
void RedChannel(char*	生成图像的红色通道图像, input 是输入文

input, char* output)	件名,output 是输出文件名。支持 24 位 BMP
	图像。
	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。
	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
	图像。
	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。
input, char* output)	<b>支之图的条件</b>
	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
1 ,	像。
	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
	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。
	缩放 (双线性插值), input 是输入文件名,
input, char* output, double	output 是输出文件名。value 是放大倍数,
value)	如 value=0.5。支持 24 位 BMP 图像。
void Translation24(char*	平移, input 是输入文件名, output 是输出
input, char* output, int x, int	文件名。x是横轴的平移量,y是纵轴的平
1 -	移量,如 x=-10, y=-30。支持 24 位 BMP 图
	像。
void Mirror24(char*	镜像, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 24 位 BMP 图像。
void Rotate24(char*	旋转, input 是输入文件名, output 是输出
input, char* output, double	文件名。degree 是旋转的度数。支持 24 位
degree)	BMP 图像。
void	给定阈值法处理图像,使图片黑白化,input
GivenThresholdMethod(char*	是输入文件名, output 是输出文件名。
input, char* output, int	threshold 是给定的阈值,如
threshold)	threshold=100。支持24位BMP图像。
void	迭代阈值法处理图像,使图片黑白化,input
IterativeThresholdMethod(cha	是输入文件名, output 是输出文件名。支持
r* input, char* output)	24 位 BMP 图像。
void	Ostu (大津法) 阈值分割, input 是输入文
OstuThresholdSegmentationMet	件名,output 是输出文件名。支持24位BMP
hod(char* input, char*	图像。
output)	
ou cpu c/	

	, 月龄山之胜夕 土柱 0.4 户 DVD 图
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
	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 Imagel 型数据,Imagel 型数据的结构
	如下:
	typedef struct
	{
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
	}Image1;
void SaveImage1(char*	将 Imagel 型数据保存为 BMP 图像, output
output, Image1* img)	是生成的 BMP 图像文件名, img 是要保存的
	图像数据。支持 8 位和 24 位 BMP 图像。
	Imagel 型数据的结构如下:
	typedef struct
	{
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
void	}Image1; 图像对比度扩展, input 是输入文件名,
ImageContrastExtension(char*	output 是输出文件名。
input, char* output, double	其中,可参考: double
m, double g1, double g2, double	m=1.5,g1=100.0,g2=200.0; m 对应斜率
a)	double $a=(255.0-m*(g2-g1))/(255.0-$
	(g2-g1));
	支持 8 位 BMP 图像。
void Binaryzation(char*	图像二值化,input 是输入文件名,output
	, , , , , , , , , , , , , , , , , , ,

input, char* output, int	是输出文件名。threshold 是将灰度值转化
threshold)	为二值的阈值,如 threshold=80。支持 24
	位 BMP 图像。
void	全局二值化,input 是输入文件名,output
GlobalBinarization(char*	是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	自适应二值化, input 是输入文件名,
AdaptiveBinarization(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	膨胀操作, input 是输入文件名, output 是
ExpansionOperation(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	腐蚀操作, input 是输入文件名, output 是
CorrosionOperation(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void Operation1(char*	开操作, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void Closed1(char*	闭操作, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void Negativel(char*	图像反色, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void Negative(char*	图像反色, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void ImageSynthesis(char*	图像合成。
input1, char* input2, char*	
output)	
void BlackWhite(char*	黑白化,支持8位和24位BMP图像。T是
input, char* output, float	阈值, border 是边界范围, 如: T=50,
T, int border)	border=0.
void Mosaic(char*	马赛克化图像, w 和 h 是输出图像的宽和
input, char* output, int w, int	高。支持 PNG 图像。
h)	
<pre>IMAGE Image_bmp_load(char*</pre>	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	日 WHJ IT IDV/II/文HJ/MC14。
input, float angle)	
IMAGE	图像的镜像翻转。
TransformShapeUpturn(IMAGE	□ 19/11 0/ 19/11/14 0
input, int a)	
void	彩色图转灰度图,对于 GrayscaleMode 的
TransformColorGrayscale(IMAG	信: 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
I im, interest expectations of the state of	量法,6表示蓝色分量法。
void	二值图(自定义阈值法)。
TransformColorBWDIY(IMAGE	
input, unsigned char	
Threshold)	
void	二值图(大津法 OSTU,适用双峰直方图。)
TransformColorBWOSTU(IMAGE	一世四(八叶四 0010, 旭/II/外"
input)	
void	二值图(三角法 TRIANGLE,适用单峰直方
TransformColorBWTRIANGLE(IMA	图。)
GE input)	12107
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	Construction
AGE input, int areaSize)	The distance and the distance of the distance
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,
                                 2, 4, 2,
                                 1, 2,1
IMAGE
                             低通滤波。
                             // 低通滤波卷积核 LP1
Wavefiltering LowPass(IMAGE
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
input, double* kernels)
                             double KERNELS Wave HighPass HP1[9] =
                                 -1, -1, -1,
                                 -1, 9, -1,
                                 -1, -1, -1
                             };
                             // 高通滤波卷积核 HP2
                             double KERNELS_Wave_HighPass_HP2[9] =
                                 0, -1, 0,
                                -1, 5, -1,
                                 0, -1, 0
```

```
};
                            // 高通滤波卷积核 HP3
                            double KERNELS Wave HighPass HP3[9] =
                                1, -2, 1,
                               -2, 5, -2,
                                1, -2,1
                            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
                            差分边缘检测。
IMAGE
EdgeDetectionDifference(IMAG
                            // 差分垂直边缘检测卷积核
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
                            double
                            KERNELS Edge difference horizontal[9]
                                0, -1, 0,
                                0, 1, 0,
                                0, 0, 0
                            };
                            // 差分垂直和水平边缘检测卷积核
                            double KERNELS Edge difference VH[9]
                               -1, 0, 0,
```

```
0, 1, 0,
                                 0, 0, 0
IMAGE
                             Sobel 边缘检测。
KernelsUseEdgeSobel(IMAGE
                             // Sobel X边缘检测卷积核
        double*
                             double KERNELS Edge Sobel X[9] =
input,
                  kernels1,
double* kernels2)
                                 -1, 0, 1,
                                -2, 0, 2,
                                 -1, 0, 1
                             };
                             // Sobel Y边缘检测卷积核
                             double KERNELS_Edge_Sobel_Y[9] =
                                -1, -2, -1,
                                 0, 0, 0,
                                 1, 2, 1
                             Laplace 边缘检测。
IMAGE
EdgeDetectionLaplace(IMAGE
                             // Laplace 边缘检测卷积核 LAP1
input, double* kernels)
                             double KERNELS_Edge_Laplace_LAP1[9] =
                                 0, 1, 0,
                                 1, -4, 1,
                                 0, 1, 0
                             };
                             // Laplace 边缘检测卷积核 LAP2
                             double KERNELS Edge Laplace LAP2[9] =
                                -1, -1, -1,
                                -1, 8, -1,
                                -1, -1, -1
                             };
                             // Laplace 边缘检测卷积核 LAP3
                             double KERNELS_Edge_Laplace_LAP3[9] =
                                -1, -1, -1,
                                -1, 9, -1,
                                -1, -1, -1
                             };
```

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

	1
	unsigned int w;
	unsigned int h;
	BGRA* color;
	}IMAGE, *PIMAGE;
	声明:
	<pre>IMAGE Image_bmp_load(char* filename);</pre>
	void Image_bmp_save(char*
	filename, IMAGE im);
	参考:
	// 用于处理
	IMAGE input2 =
	<pre>Image_bmp_load(inputfile);</pre>
	// 用于保存
	IMAGE input2=
	<pre>Image_bmp_load(inputfile);</pre>
	input2=FaceDetection(input1,input2,KE
	<pre>RNELS_Wave_Average);</pre>
	// 保存图片
	<pre>Image_bmp_save(outputfile, input2);</pre>
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
Court I to rome, cited they/	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
input, String Output)	果文件。
void HorizontalMirror(char*	水平镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持8位BMP图像。
void MirrorVertically(char*	垂直镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持8位BMP图像。
void XMirroring(char*	X 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持8位BMP图像。
void YMirroring(char*	Y 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持8位BMP图像。
1,	->-11 H - >-44 - H H W -

void ImageConvolution(char*	图像卷积, input 是输入文件名, output 是
input, char* output, double**	输出文件名。Kernel 是卷积核,如 double
Kernel, int n, int m)	$[Kernel[3][3]] = \{\{-0.225, -0.225-\}\}$
	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	
radius)	
void SpatialGaussFiter(char*	空间高斯过滤器,参考: radius=3。
input, char* output, int	
radius)	
void	空间统计滤波器,参考: radius=3, T=0.2。
SpatialStatisticalFiter(char	
* input, char* output, int	
radius, float T)	
void FFTAmp(char*	FFT 放大器,参考: inv=false。
input, char* output, bool inv)	
void FFTPhase(char*	FFT 相位,参考: inv=false。
input, char* output, bool inv)	
void STDFT1(char*	参考: inv=false。
input, char* output, bool inv)	
void STDFT2(char*	参考: inv=false。
input, char* output, bool inv)	
void SpectrumShaping(char*	图像频域滤波, FFT 变换相位谱,
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

Cross Done is in m94 (DMDM at what	(上:nout 十小担目)
CrossDenoising24(BMPMat**	(与 input 大小相同)。
input, BMPMat** output, BMPMat	
threshold, BMPMat target)	图像土於甘油佈妻 out nut 用工机方灶用
void	图像去除某种像素, output 用于保存结果
CrossDenoising8 (unsigned	(与 input 大小相同)。
char** input, unsigned char**	
output, unsigned char	
threshold, unsigned char	
target)	图格士法 / 1 1) 目标形法法反码士工会
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)	圆块土层 / 1 1) 目标式层速点处土工车
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)	<b>国体操儿</b> , 日松 ) 之体力 , 日
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)	11011 FW /I. 42 #V 11 5 400
void SharpenUSM(char*	USM 锐化,参考: radius=5, amount=400,
input, char* output, int	threshold=50。
radius, int amount, int	
threshold)	大OA P DUD 图像(译),如今华京 人
	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 have	形线框的红色分量, green 是矩形线框的绿     免公量   blue 見短形的蓝色公量
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 Ing2/ImagaCanaration (about	JPG 图像生成, filename 是生成的 JPG 图像     文件名 width 是图像的第一hoight 是图像
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)	具近郊场估法土岬枚 :
void ImageScalingNearestNeighborI	最近邻插值法去栅格,input 是输入文件
ImageScalingNearestNeighborI	名,output 是输出文件名。lx 和 ly 是长和
nterpolation(char*	宽需要缩放的倍数。支持 8 位 BMP 图像。

input, char* output, float	
lx, float ly)	
	四张州长佐江土州牧 : 目於入文件
void	双线性插值法去栅格, input 是输入文件
ImageScalingBilinearInterpol	名,output 是输出文件名。1x 和 1y 是长和
ation(char* input, char*	宽需要缩放的倍数。支持8位BMP图像。
output, float 1x, float 1y)	
void	双线性插值, input 是输入文件名, output
BilinearInterpolationScaling	是输出文件名。ExpScalValue 是期望的缩
(char* input, char*	放倍数 (允许小数)。支持 BMP 图像。
output, float ExpScalValue)	
void	最近邻插值,input 是输入文件名,output
NearestNeighborInterpolation	是输出文件名。ExpScalValue 是期望的缩
Scaling(char* input, char*	放倍数 (允许小数)。支持 BMP 图像。
output, float ExpScalValue)	
void ZoomImg(unsigned char	二次线性插值图像缩放。
*input, unsigned char	
*output, int sw, int sh, int	
channels, int dw, int dh)	
void	图像修复, output 用于保存结果 (与 input
CrossDenoising24(BMPMat**	大小相同), target 是污点像素, weight 是
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	少久以至水纵。
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)	口。 又的 o 匹 Dim 国 图 , 四 工
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	
12, inc	围绕的中心点的坐标,color 是旋转后非原
color)	围绕的中心点的坐标, color 是旋转后非原图区域的填充颜色。
color)	图区域的填充颜色。
color)	图区域的填充颜色。
color) void Rotation24(char*	图区域的填充颜色。 图像旋转, input 是输入文件名, output 是

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
angle, unsigned char color)	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	NA PROPERTY OF THE PROPERTY OF
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	
a, float b)	
void NoiseExp(char*	指数噪声,参考: a=0.1。
input, char* output, float a)	
void NoiseImpulse(char*	椒盐噪声,参考: a=0.2, b=0.2。
input, char* output, float	
a, float b)	
void grayToColor(FILE*	灰色转伪彩色,input 是输入文件,output
input,FILE* output)	是输出文件。支持 8 位和 24 位 BMP 图像。
void ImageThinning(char*	图像细化, input 是输入文件名, output 是
input, char* output, char**	输出文件名。支持 4 位 BMP 图像。n 是 str
str, int n, int ml, int a, int b)	第一维的大小, m1 是第二维的大小, 形如
	str[n][m1]; a和b是相关的调节参数,可
	以为 a=3, b=5。
	参考模板:
	char $str[6][8] = \{ \{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, $
	0, 0, }, { 255, 0, 255, 0, 0, 255, 0,
	0 },
	$\{255, 0, 255, 255, 0, 255, 0, $
	255 }, { 255, 255, 255, 0, 0, 255,
	255, 255 },
	{ 255, 0, 255, 255, 0, 255, 255,
	255 }, { 0, 255, 255, 255, 255, 255,
	255, 255 } };
int	返回图像像素的最小值, filename 是输入
MinimumValueOfImagePixels(ch	的图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	
int	返回图像像素的最大值,filename 是输入
MaximumValueOfImagePixels(ch	的图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	
float	返回图像像素的均值, filename 是输入的
AverageValueOfImagePixels(ch	图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	
double	返回图像像素的标准差,filename 是输入
StandardDeviationOfImagePixe	的图像文件名。支持 8 位和 24 位 BMP 图像。
ls (char* filename)	
double EntropyOfImage(char*	返回图像的熵,支持8位和24位BMP图像。
filename)	
float*	filename 是输入的图像文件名。存储每个
CountTheFrequencyOfPixels(ch	像素的频率,像素值为 0 <sup>2</sup> 55, 返回值数组
ar* filename)	中的元素序号即为像素值,该序号在数组下
ar rirename)	的值即为这个像素的频率。支持 8 位和 24
	位 BMP 图像。
	T Duff

void Rotate(char*	_ · · · · · · · · · · · · · · · · · · ·
input, char* output, int	interpolation=0或interpolation=1。
angle, int interpolation)	
void HSV(char* input, char*	图像色调饱和度明度调节,参考: h=120,
output, int h, int s, int v)	s=60, v=20。
void 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	和国版版,多· 5. IddId5 10, 5mooth 100。
radius, int smooth)	
void HaloFilter(char*	晕角滤镜,参考: ratio=100。
	年用
input, char* output, int	
ratio)	大座古子园 台北 1mm111 050
void GrayHistogram(char*	灰度直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void RedHistogram(char*	红色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void GreenHistogram(char*	绿色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void BlueHistogram(char*	蓝色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization2(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output, int	BMP 图像。imgBit 是输入图像的位数。
imgBit)	
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
	Output 定制出文件石。文符 6 位和 24 位     BMP 图像。
input, char* output)	
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,
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)	
void ToRIM(char* input, char*	一般图像转到 RIM 图像,支持 PNG、JPG 和
output)	TGA 图像。
void ToImage(char*	RIM 图像转到一般图像,支持 PNG、JPG 和
input, char* output, int	TGA 图像。jpg_quality=25。
jpg_quality)	Ton Editor Japanese J
void	将 1 位深度的单色 BMP 图片转成热敏打印
ImprimanteThermique(char*	机的位图打印输出,支持的热敏打印机的位
input, char* output, ARRAY3	图打印指令为\strong\ESC *\strong\指
skip_cmd, unsigned short	令。
PRINTER_TYPE_BMP, unsigned	typedef unsigned char ARRAY3[3];参考:
char mode, unsigned int	output="output.pbin", skip cmd =
FILE_TYPE_AD, unsigned char	{0x1B, 0x4A, 0x00}, PRINTER TYPE BMP 是
a, unsigned char b)	打印机位图打印指令码标识,
,	PRINTER TYPE BMP=(0x2A1B), mode 是打印
	机位图打印模式, mode=33, FILE_TYPE_AD
	是图片类型, "AD"表示广告图片,
	FILE_TYPE_AD=(0x4441), a=0x80, b=1.
void WhiteBalance(const	白平衡。
char* input, const char*	
output)	
void Sobel(char* input, char*	Sobel 算子, magnScale=0.35,
output, double	threshold=130。支持 PGM 和 PBM 图像。
magnScale, double threshold)	
void Canny(char* input, char*	Canny 算子, magnScale=0.35 ,
output, double	lowThreshold=55, highThreshold=120。支
magnScale, double	持 PGM 和 PBM 图像。
lowThreshold, double	
highThreshold)	
void BlackWhite(char*	黑白化,threshold=100,background=0。
input, char* output, int	支持 PGM 和 PBM 图像。
threshold, int background)	
void	区域连通,threshold=100,background=0,
ConnectedComponents(char*	threshold1=100。支持 PGM 和 PBM 图像。
input, char* output, int	
threshold, int background, int	
threshold1)	
void CleanImage(char*	清洁图像。支持 PGM 和 PBM 图像。
input, char* output)	
void   NoiseImage(char*	噪声化图像,probability=0.1。支持 PGM
input, char* output, float	和 PBM 图像。
probability)	TELLA MIN
void	圆检测。scale1=1,gamma1=1.0,
HoughTransformCircle1(char*	magnScale=0.5 , lowThreshold=85 ,

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

highThreshold=150, scale=0, gamma=1.0, sigma 和 kernelSize 用于平滑的高斯 5x5 内核, sigma=1.0, kernelSize=5。

如果 scale==0,则值保持不变,但如果它们低于 0 或高于 255,则它们分别设置为 0 或255。

如果 scale! =0,则对值进行缩放,使得最小值为零,最大值为 255。

设置 gamma 值允许指数缩放, 启用 gamma==1.0。

支持 PGM 和 PBM 图像。

void

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

圆 检 测 。 scale1=1 , gamma1=1.0 , magnScale=0.5 , lowThreshold=85 , highThreshold=150, scale=0, gamma=1.0, sigma 和 kernelSize 用于平滑的高斯 5x5 内核,sigma=1.0,kernelSize=5,number=10 表示图像的目视检查有 10 个圆圈, minDist=35。

如果 scale==0,则值保持不变,但如果它们低于 0 或高于 255,则它们分别设置为 0 或255。

如果 scale! =0,则对值进行缩放,使得最小值为零,最大值为 255。

设置 gamma 值允许指数缩放,启用gamma==1.0。

支持 PGM 和 PBM 图像。

double\*\*

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

圆检测。scale1=1,gamma1=1.0,magnScale=0.5,lowThreshold=85,highThreshold=150,scale=0,gamma=1.0,sigma和kernelSize用于平滑的高斯5x5内核,sigma=1.0,kernelSize=5,number=10表示图像的目视检查有10个圆圈,minDist=35。

如果 scale==0,则值保持不变,但如果它们低于 0 或高于 255,则它们分别设置为 0 或 255。

如果 scale! =0,则对值进行缩放,使得最小值为零,最大值为 255。

设置 gamma 值允许指数缩放, 启用 gamma==1.0。

返回以(vCenter, hCenter)和半径(vRadius, hRadius)为中心的椭圆数据,共有 number 组数据,每组包含一个椭圆数据,第一个元素是 vCenter,第二个元素是

	hCenter,第三个元素是 vRadius,第四个元
	素是 hRadius。
	支持 PGM 和 PBM 图像。
Void Shana Edga Data atian 1 (aharu	形状边缘检测,CANNY_THRESH4=35, CANNY blur4=7。支持PNG图像。
ShapeEdgeDetection1(char*	CANNI_DIUI4-7。 文持 FNG 图像。
input, char* output, unsigned char CANNY THRESH4, int	
CANNY blur4)	
void	形状边缘检测,CANNY_THRESH4=35,
ShapeEdgeDetection2(char*	CANNY_blur4=7。支持 PNG 图像。
input, char* output, unsigned	CUMI_pini4-1。 太祖 i no 国像。
char CANNY THRESH4, int	
CANNY blur4)	
void	形状边缘检测, CANNY_THRESH=50,
ShapeEdgeDetection3(char*	CANNY BLUR=12。支持 PNG 图像。
input, char* output, unsigned	
char CANNY THRESH, int	
CANNY_BLUR)	
void	形状边缘检测, CANNY_THRESH=50,
ShapeEdgeDetection4(char*	CANNY_BLUR=12。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH, int	
CANNY_BLUR)	
void	形状边缘检测,CANNY_THRESH2=10,
ShapeEdgeDetection5(char*	CANNY_BLUR2=2。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH2, int	
CANNY_BLUR2)	
void	形状边缘检测,CANNY_THRESH2=10,
ShapeEdgeDetection6(char*	CANNY_BLUR2=2。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH2, int	
CANNY_BLUR2)	形
Void ShanoFidgoDataction7(char*	形 状 边 缘 检 测 , CANNY_THRESH3=45 , CANNY blur3=10。支持 PNG 图像。
ShapeEdgeDetection7(char* input, char* output, unsigned	CVIMMI_DIUI 2-10。 文书 LING 園像。
char CANNY_THRESH3, int	
CANNY_blur3)	
void	形状边缘检测,CANNY_THRESH3=45,
ShapeEdgeDetection8(char*	CANNY blur3=10。支持 PNG 图像。
input, char* output, unsigned	
char CANNY THRESH3, int	
CANNY_blur3)	
<u> </u>	

## 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。

## 高级算子

void	Blob 分析, c1 和 c2 是与颜色相关的参数,参考:
BlobAnalysis(char*	c1=128, c2=127。支持 BMP 图像。
input, char*	
output, int c1, int c2)	
void	Blob 分析, c1 和 c2 是与颜色相关的参数,参考:
BlobAnalysis1(char*	c1=128, c2=127。支持 BMP 图像。
input, char*	
output, int c1, int c2)	
void	验 证 码 生 成 。
VerificationCodeGene	sigma=10, noise type=2, a=10, b1=128, b2=127,
ration(char*	b3=2, b4=8, b5=12, b6=0, b7=1, b8=-100, b9=-100,
inputText, char*	b10=1, b11=3, b12=6, int b13=40, foint=30, num
output, int num, int	是字符数量,depth=1,spectrum=3,shared=0。
foint, int a, int	
b1, int b2, int b3, int	
b4, int b5, int b6, int	
b7, int b8, int b9, int	
b10, int b11, int	
b12, int b13, double	
sigma, unsigned int	
noise_type, int	
width, int height, int	
depth, int	
spectrum, bool shared)	
void	角点检测,threshold=10000,k=0.06,sigma=1.0,
CornerDetection(char	width=640, height=480, channels=1。支持 PNM 图
* input, char*	像。
output, float	
threshold, float k,	
float sigma, int	
width, int height,	
int channels)	

```
vector < Keypoint >
                      角点检测,返回角点数据。threshold=10000,
CornerDetection1(cha
                      k=0.06, sigma=1.0, width=640, height=480,
                      channels=1。支持 PNM 图像。
         input, char*
                      需引入以下结构体
output, float
threshold,
                      typedef struct {
           float k,
float
           sigma, int
                         float x;
width.
        int
             height,
                         float y:
int channels)
                         float score;
                      } Keypoint;
vector < Corner:: Keypo
                      角点检测,返回角点数据。threshold=2000,k=1,
int>
                      sigma=1.2。支持 PNM 图像。
CornerDetection(char
                      需引入以下命名空间:
* input, int width, int
                     namespace Corner {
height, int
                      struct Keypoint {
channels, float
                             float x;
threshold,
          float k,
                             float y:
float sigma)
                             float score;
                      };
void Structure(char*
                     特征归一化统计,参考: sigma=2。支持多种图像格
input, char*
                      式。
output, float sigma)
                     角点检测,参考: sigma=2, method=0。支持多种图
void Cornerness (char*
input, char*
                      像格式。
output, float
sigma, int method)
void
       Corners (char*
                      角点检测,参考: sigma=2, thresh=0.4, window=5,
input, char*
                      nms=3, corner method=0。支持多种图像格式。
output, float
              sigma,
float
       thresh,
                 int
window, int nms, int
corner method)
                     特征匹配,参考: thresh3=5, k=10000, cutoff=50,
void FindMatch(char*
input1, char*
                      thresh4=5, sigma=2, thresh=0.4, window=5, nms=3,
                      corner method=0, sigma1=2, thresh1=0.4,
input2, char*
output, float thresh3,
                      window1=5, nms1=3, corner method1=0, sigma2=2,
int
         k,
                      thresh2=0.4
                                      window2=5
                                                       nms2=3
                 int
cutoff, float
                      corner method2=0, sigma5=2, thresh5=0.4,
                     window5=7, nms5=3, corner method5=0, sigma6=2,
thresh4, float sigma,
       thresh,
                      corner method6=0, thresh6=0.3, window6=7,
float
                 int
window, int nms, int
                      nms6=3, inlier thresh6=5, iters6=1000,
corner method, float
                      cutoff6=50, acoeff6=0.5。支持多种图像格式。
sigmal,
               float
threshl, int windowl,
```

```
int
       nms1,
                int
corner method1, float
sigma2,
               float
thresh2, int window2,
int
       nms2,
corner method2, float
sigma5,
thresh5, int window5,
int
       nms5.
                 int
corner method5, float
sigma6,
corner method6, float
thresh6, int window6,
      nms6.
inlier_thresh6,
                 int
iters6, int cutoff6,
float acoeff6)
vector (Descriptor)
                     角点检测,返回检测结果。参考: sigma1=2,
                                      window1=5,
HarrisCorner(char*
                     thresh1=0.4,
                                                      nms1=3
input, char*
                     corner method1=0。支持多种图像格式。
output, float sigmal,
                     需引入以下结构体:
float threshl,
                 int
                     struct Point {
windowl, int nms1.
                       double x, y;
int corner method1)
                       Point(): x(0), y(0) {}
                       Point (double x, double y) : x(x), y(y) {}
                     };
                     struct Descriptor {
                       Point p;
                       vector<float> data;
                       Descriptor() {}
                       Descriptor(const Point& p) : p(p) {}
vector < Match >
                     描述匹配项,返回描述结果。参考: sigma1=2,
MatchDescriptors(cha
                     thresh1=0.4, window1=5,
                                                      nms1=3
        input1, char*
                     corner method1=0, sigma2=2, thresh2=0.4,
input2, char*
                     window2=5, nms2=3, corner_method2=0。支持多种
output, float sigmal,
                     图像格式。
float threshl,
                     需引入以下结构体:
                 int
windowl, int nms1,
                     struct Point {
int
                       double x, y;
corner_method1, float
sigma2,
               float
                       Point(): x(0), y(0) {}
```

```
Point (double x, double y) : x(x), y(y) {}
thresh2, int window2,
int
       nms2,
                  int
corner method2)
                       struct Descriptor {
                         Point p;
                         vector<float> data;
                         Descriptor() {}
                         Descriptor(const Point& p) : p(p) {}
                       struct Match {
                         const Descriptor* a=nullptr;
                         const Descriptor* b=nullptr;
                         float distance=0.f;
                         Match() {}
                         Match (const Descriptor* a, const Descriptor*
                       b, float
                                  dist=0.f
                                             :
                                                      a (a),
                                                               b(b),
                       distance(dist) {}
                         bool operator<(const Match& other) { return</pre>
                       distance<other.distance; }</pre>
                       };
void
                       绘制角点,参考: thresh3=5, k=10000, cutoff=50,
DrawInliers(char*
                       thresh4=5, sigma=2, thresh=0.4, window=5, nms=3,
input1, char*
                       corner method=0, sigma1=2, thresh1=0.4,
input2, char*
                       window1=5, nms1=3, corner method1=0, sigma2=2,
output, float thresh3,
                       thresh2=0.4
                                   ,
                                        window2=5,
                                                         nms2=3,
int
         k.
                  int
                       corner method2=0, sigma5=2, thresh5=0.4,
cutoff, float
                       window5=7, nms5=3, corner_method5=0, sigma6=2,
                       corner method6=0, thresh6=0.3, window6=7,
thresh4, float sigma,
                       nms6=3, inlier thresh6=5, iters6=1000,
float
       thresh,
                  int
window, int nms, int
                       cutoff6=50, acoeff6=0.5。支持多种图像格式。
corner method, float
sigmal,
                float
thresh1, int window1,
int
       nms1,
                  int
corner_method1, float
sigma2,
               float
thresh2, int window2,
       nms2,
int
corner method2, float
sigma5.
thresh5, int window5,
```

int

nms5.

int

corner\_method5, float sigma6, int corner\_method6, float thresh6, int window6, int nms6, float inlier\_thresh6, int iters6, int cutoff6, float acoeff6)

void

PanoramaImage(char\* input1, char\* input2, char\* output, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int corner method1, float sigma2, thresh2, int window2, nms2, int corner method2, float sigma5, float thresh5, int window5, int nms5, corner method5, float sigma6, int corner method6, float thresh6, int window6, int nms6. float inlier\_thresh6, int iters6, int cutoff6, float acoeff6)

制造全景图像,参考: thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner method=0, sigma1=2, thresh1=0.4, window1=5 nms1=3, corner method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner method2=0, sigma5=2, thresh5=0.4 window5=7 nms5=3corner method5=0, sigma6=2, corner method6=0, window6=7, thresh6=0.3, nms6=3inlier thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5。支持多种图像格式。

void
Cylindrical(char\*
input1, char\*
input2, char\*
output, float f1, float
f2, float thresh3, int

角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner\_method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3, corner\_method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner\_method2=0, sigma5=2,

k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float threshl, int windowl, int nms1, int corner method1, float sigma2, float thresh2, int window2, int nms2, corner method2, float sigma5, thresh5, int window5, int nms5, corner method5, float sigma6, corner method6, float thresh6, int window6, int nms6, float inlier\_thresh6, int iters6, int cutoff6, acoeff6, float float sigma7, int corner method7, float thresh7, int window7, int nms7, float inlier thresh7, int iters7, int cutoff7, float acoeff7)

thresh5=0.4 , window5=7 , nms5=3 , corner\_method5=0, sigma6=2, corner\_method6=0, thresh6=0.3 , window6=7 , nms6=3 , inlier\_thresh6=5 , iters6=1000 , cutoff6=50 , acoeff6=0.5 , sigma7=2 , corner\_method7=0 , thresh7=0.3 , window7=7 , nms7=3 , inlier\_thresh7=5 , iters7=1000 , cutoff7=50 , acoeff7=0.5 。 支持多种图像格式。

void Spherical(char\* input1, char\* input2, char\* output, float f1, float f2, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh. int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int

角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner\_method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3, corner\_method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner method2=0, sigma5=2, window5=7 thresh5=0.4 nms5=3corner method5=0, sigma6=2, corner method6=0, thresh6=0.3 , window6=7 nms6=3 inlier\_thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5, sigma7=2, corner method7=0, nms7=3thresh7=0.3, window7=7inlier\_thresh7=5, iters7=1000, cutoff7=50,

.1 11 01	
corner_method1, float	acoeff7=0.5。支持多种图像格式。
sigma2, float	
thresh2, int window2,	
int nms2, int	
corner_method2, float	
sigma5, float	
thresh5, int window5,	
int nms5, int	
corner_method5, float	
sigma6, int	
corner_method6, float	
thresh6, int window6,	
int nms6, float	
inlier_thresh6, int	
iters6, int cutoff6,	
float acoeff6, float	
sigma7, int	
corner method7, float	
thresh7, int window7,	
int nms7, float	
inlier_thresh7, int	
iters7, int cutoff7,	
float acoeff7)	
int* FindLine(char*	直线检测,返回直线的 theta 和 rho,支持 RAW 图
input, char*	像。
output, int width, int	[3 <b>\</b> ○
height)	
int* FindCircle(char*	圆检测,返回圆心的坐标和圆的半径,支持 RAW 图
input, char*	像。sigma=1.4,tmin=70,tmax=150。
output, int width, int	gro Signa 1. 1, thill 10, than 100.
height, float sigma,	
int tmin, int tmax)	
void	模板匹配, a=0.5, b=0, a1=0.5, b1=0.5, c=0.2,
TemplateMatching(cha	ps 是相似度,如 ps=0.5。支持 24 位 BMP 图像。
r* input, char*	ps 足作的友,如 ps-0.0。 文寸 2年世 DMI 国家。
= .	
Template, char* output, unsigned int	
output, unsigned int b, double ps, double	
a, double al, double	
bl, double c)	
SearchResult	模板匹配,返回目标位置的左上角坐标和相似值。
TemplateMatching(cha	模倣匹配,返回日你位直的左工用坐你和相似值。 支持 PNG 图像。
	黑引入以下结构体:
	而引入以下结构体: struct SearchResult {
Template)	Struct Searchiesurt )

	• ,
	int x, y;
	double value;
	};
SearchResult	模板匹配,返回目标位置的左上角坐标和相似值。支
TemplateMatching(uin	持 PNG 图像。
t8 t* input, uint8 t*	需引入以下头文件:
Template, int	#define STB_IMAGE_IMPLEMENTATION
imgWidth, int	#include "stb_image.h"
imgHeight, int	#include <cstdint></cstdint>
imgBpp, int	#include <complex></complex>
patWidth, int	#include <vector></vector>
patHeight, int patBpp)	参考例程:
	int imgWidth, imgHeight, imgBpp;
	int patWidth, patHeight, patBpp;
	uint8 t* input = stbi load(inputfile,
	&imgWidth, &imgHeight, &imgBpp, 3);
	uint8_t* Template = stbi_load(Templatefile,
	&patWidth, &patHeight, &patBpp, 3);
void	图像标定。
ImageCalibration1(ch	
ar* input, char*	
output)	
std::vector <line></line>	图像标定。
ImageCalibration2(ch	需引入以下结构体:
ar* input, char*	struct Line {
output)	double m, b;
output)	, i
	int dist_o; // distance to origin point
	when
	<pre>// line(x = dist_o) is perpendicular</pre>
	to x axis
	int x0, x1, y0, y1; // two end points
	<pre>int end_point_num;</pre>
	Line(double _m, double _b, int _dist_o =
	0,  int  x0 = 0,  int  y0 = 0,
	int x1 = 0, int y1 = 0, int
	end point num = 0)
	: m(_m), b(_b), dist_o(_dist_o),
	x0(_x0), x1(_x1), y0(_y0), y1(_y1),
	end_point_num(_end_point_num) {}
	};
std::vector	圆检测,返回检测结果。size1=5,size2=5,size3=7。
<centerspoint></centerspoint>	支持 BMP 图像。
FindCircles(char*	需引入以下结构体:
input, char*	struct Point {
_ * ′	

output, int size1, int size2, int size3)	<pre>Point(int x = 0, int y = 0) { this-&gt;x = x; this-&gt;y = y; }   int x;   int y; }; struct CentersPoint {     CentersPoint(Point point, int radius) { this-&gt;point = point; this-&gt;radius = radius; count = 1; }     Point point;</pre>
	<pre>int count; int radius; };</pre>
<pre>void ImgDisplay(char* input, char* WindowsName)</pre>	图像显示, input 是输入的图像文件名, Windows Name 是要创建的窗口名称。
<pre>vector<int> HoughTransform1 (char * input, char* output, unsigned char color[3], int numOfLines, int fillGap, int minLength)</int></pre>	霍夫直线检测,返回检测结果。如果共线线段之间的间隙小于 fillGap,则连接它们,如果合并的行短于minLength,则放弃它,numOfLines 是可能的线段数量,numOfLines=10,fillGap=5,minLength=60,color={0,255,0}。支持8位BMP图像。
<pre>vector<int> HoughTransform2(char * input, char* output, unsigned char color[3], int numOfLines, int fillGap, int minLength)</int></pre>	霍夫直线检测,返回检测结果。如果共线线段之间的间隙小于 fillGap,则连接它们,如果合并的行短于minLength,则放弃它,numOfLines 是可能的线段数量,numOfLines=10, fillGap=5, minLength=60, color={ 0, 255, 0 }。支持 8 位 BMP 图像。
void SobelGradient(char* input, char* output)	用 sobel 算子分别求 x 方向, y 方向的梯度。
void SobelNoneMaxSpress(c har* input, char* output)	用 sobel 算子分别求 x 方向, y 方向的梯度, 再用梯度方向和非极大值抑制去掉冗余信息。
void HytheresisThresholdi ng(char* input, char* output, double high, double low)	Hythereis 阈值处理,high=130,low=60。

void	霍夫变换,high=130, low=60, threshold=0.4。
HoughTransfer1(char*	在人文队,High 100,10w 00,threshold 0.16
input, char* output,	
double high, double	
low, double	
threshold)	
void	霍夫变换,high=130,low=60,threshold=0.4。
HoughTransfer3(char*	在人文沃,IIIgli-150,10w-00,tili esiloiu-0.4。
input, char* output,	
double high, double	
low, double	
threshold)	
vector <int></int>	
HoughTransfer2(char*	在入文揆, 返回用点信息, 以两千九条/) 组。   high=130, low=60, threshold=0.4。
_	nigh-150, fow-60, threshold-0.4.
input, double high, double low, double	
,	
threshold)	4±.4T ±0.5ml
void	特征检测。
FeatureDetection1(ch	
ar* input, char*	
output)	<u> </u>
std::vector <sift::ke< td=""><td>特征检测。</td></sift::ke<>	特征检测。
ypoint>	需引入以下空间:
FeatureDetection2(ch	namespace sift
ar* input, char*	{
output)	struct Keypoint {
	//离散坐标系
	int i;
	int j;
	int octave;
	int scale; //倍频程内高斯图像的索引
	//连续坐标(内插)
	float x;
	float y;
	float sigma;
	float extremum_val; //插值 DoG 极值值
	std::array <uint8_t, 128=""> descriptor;</uint8_t,>
	};
void	特征匹配。支持 PGM 图像。
FeatureMatching(char	
* input1, char*	

input2, char* output)	
std::vector <std::pai< td=""><td>特征匹配,返回匹配结果。</td></std::pai<>	特征匹配,返回匹配结果。
	付征匹乱,及巴匹乱组术。
· ·	
FeatureMatching1(cha	
r* input1, char*	
input2, char* output)	
std::vector <std::pai< td=""><td>特征匹配,返回匹配结果。</td></std::pai<>	特征匹配,返回匹配结果。
r <int, int="">&gt;</int,>	
FeatureMatching2(cha	
r* input1, char*	
input2, char* output)	
void Canny (char*	Canny 算子,至少支持 JPG 图像,input 是输入文件
input, char*	名,output 是输出文件名,参考:lowThreshold=50,
output, int	highThreshold=150。
lowThreshold, int	
highThreshold)	
void Canny(string	Canny 算子,参考: output="output"。支持 BMP 文
input, string output)	件。
void Canny(string	Canny 算子,参考: sigma=6.0, threshold=3.5。支
input, char*	持 BMP 文件。
output, float sigma,	
float threshold)	
void Hough(char*	霍夫变换,参考: sigma=6.0, threshold=3.5,
input, char* output,	thre val=0.5, color 用于设置画出的标定点和线的
float sigma, float	颜色,如:color[3]={0,0,255}。支持BMP文件。
threshold, double	(a, a, 100) (b, 2, 100) (c, 2, 100)
thre_val, unsigned	
char* color)	
void DES_Encrypt(char	DES 加密函数,支持多种文件。PlainFile 是原文件
*PlainFile, char	的文件名,Key 是密钥字符,CipherFile 是加密后
*Key, char	的文件名,Key 定品的子科,Cipheri Tie 定加品加 的文件名。
*CipherFile)	17文件石。
	DES 解密函数,支持多种文件。CipherFile 是已加
void DES_Decrypt (char	
*CipherFile, char	密文件的文件名,Key 是密钥字符,PlainFile 是解
*Key, char *PlainFile)	密后的文件名。
int*	模板匹配,input 是母本图像,Template 是样本图像。
TemplateMatch(char*	像, output 是结果图像文件名。channels 是图像的
input, char*	像素通道数,返回值数组中的第一个元素是最大匹
Template, char*	配分数,第二个元素是目标的 X 轴坐标值,第三个
output, int	元素是目标的Y轴坐标值。至少支持PNG图像。参
channels, int	考: channels=3, ROTATION=360 或 ROTATION=1。
ROTATION)	
int*	模板匹配,input 是母本图像,Template 是样本图
TemplateMatch(image	像,output 是结果图像文件名。channels 是图像的

input, image	像素通道数,返回值数组中的第一个元素是最大匹
Template, char*	配分数,第二个元素是目标的 X 轴坐标值,第三个
output, int	元素是目标的 Y 轴坐标值。至少支持 PNG 图像。参
channels, int	考: channels=3,ROTATION=360 或 ROTATION=1。
ROTATION)	需引入以下头文件和结构体:
	#include "stb_image.h"
	#include "stb_image_write.h"
	typedef struct imageContainer {
	int x, y, n;
	unsigned char *data;
	} image;
	参考:
	image input, Template;
	<pre>input.data = stbi_load(inputFile, &amp;input.x,</pre>
	&input.y, &input.n, 3); //3 表示图像有 3 个像
	素通道
	<pre>Template.data = stbi_load(templateFile,</pre>
	&Template.x, &Template.y, &Template.n, 3);
int*	模板匹配, input 是母本图像, Template 是样本图
TemplateMatch(char*	像。channels 是图像的像素通道数,返回值数组中
input, char*	的第一个元素是最大匹配分数,第二个元素是目标
Template, int	的 X 轴坐标值,第三个元素是目标的 Y 轴坐标值。
channels, int	至少支持PNG图像。参考:channels=3,ROTATION=360
ROTATION)	或 ROTATION=1。
int*	模板匹配, input 是母本图像, Template 是样本图
TemplateMatch(image	像。返回值数组中的第一个元素是最大匹配分数,第
input, image	│二个元素是目标的 X 轴坐标值,第三个元素是目标 │
Template, int	的 Y 轴坐标值。至少支持 PNG 图像。参考:
ROTATION)	ROTATION=360 或 ROTATION=1。
	需引入以下头文件和结构体:
	#include "stb_image.h"
	#include "stb_image_write.h"
	typedef struct imageContainer {
	int x, y, n;
	unsigned char *data;
	image;
	参考:
	image input, Template;
	input.data = stbi_load(inputFile, &input.x,
	&input.y, &input.n, 3); //3 表示图像有 3 个像 素通道
	Template.data = stbi_load(templateFile,
	&Template.x, &Template.y, &Template.n, 3);
void KMeans(string	K-Means 聚类,input 是输入文件名,Clusters 是聚

input, unsigned int	类的种类数目, output 是输出文件名。支持 BMP 文
Clusters, char*	件。
output)	
void PGMSobel(char*	Sobel 算子, input 是输入文件名, output 是输出文
input, char*	件名。支持 P5 格式的 PGM 文件。
output, int	参考模板:
Mx[3][3], int	int $Mx[3][3] = \{\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, -1\}\}$
My[3][3], int max, int	0, 1}}
min)	int $My[3][3] = \{\{-1, -2, -1\}, \{0, 0, 0\}, \{1, -2\}\}$
	2, 1}}
	int max = -9999
	int min = 9999
void PGMSobelX(char*	X 方向滤波, input 是输入文件名, output 是输出文
input, char*	件名。支持 P5 格式的 PGM 文件。
output, int	参考模板:
Mx[3][3], int	int $Mx[3][3] = \{\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, \}\}$
My[3][3], int max, int	0, 1}}
min)	int $My[3][3] = \{\{-1, -2, -1\}, \{0, 0, 0\}, \{1, -2\}\}$
	2, 1}}
	int max = $-9999$
	int min = 9999
void PGMSobelY(char*	Y 方向滤波, input 是输入文件名, output 是输出文
input, char*	件名。支持 P5 格式的 PGM 文件。
output, int	参考模板:
Mx[3][3], int	int $Mx[3][3] = \{\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, \}\}$
My[3][3], int max, int	0, 1}}
min)	int $My[3][3] = \{\{-1, -2, -1\}, \{0, 0, 0\}, \{1, -2\}\}$
	2, 1}}
	int max = $-9999$
	int min = 9999
void PGMSobell(char*	Sobel 算子, input 是输入文件名, output 是输出文
input, char*	件名。min 和 max 是图像归一化相关参数,如 min =
output, int min, int	1000000, max = 0; mx 和 my 分别是 Sobel 算子的 X
max, int $mx[3][3]$ , int	和 Y 方向模板。支持 P2 和 P5 格式的 PGM。
my[3][3])	参考模板:
	$int mx[3][3] = \{$
	$\{-1, 0, 1\},$
	$\{-2, 0, 2\},\$
	{-1, 0, 1}
	}; ;
	$int my[3][3] = {$
	$\{-1, -2, -1\},\$
	$\{0, 0, 0\},\$
	{1, 2, 1}

}; void PGMSobelX1(char\* X 方向梯度, input 是输入文件名, output 是输出文 件名。min 和 max 是图像归一化相关参数,如 min = input, char\* output, int min, int 1000000, max = 0; mx 和 my 分别是 Sobel 算子的 X max, int mx[3][3], int 和 Y 方向模板。支持 P2 和 P5 格式的 PGM。 my[3][3]参考模板: int  $mx[3][3] = {$  $\{-1, 0, 1\},\$  $\{-2, 0, 2\},\$  $\{-1, 0, 1\}$ };  $int my[3][3] = {$  $\{-1, -2, -1\},\$  $\{0, 0, 0\},\$  $\{1, 2, 1\}$ }; Y方向梯度, input 是输入文件名, output 是输出文 void PGMSobelY1(char\* input, char\* 件名。min 和 max 是图像归一化相关参数,如 min = 1000000, max = 0; mx 和 my 分别是 Sobel 算子的 X output, int min, int max, int mx[3][3], int 和 Y 方向模板。支持 P2 和 P5 格式的 PGM。 my[3][3]参考模板:  $int mx[3][3] = {$  $\{-1, 0, 1\},\$  $\{-2, 0, 2\},\$  $\{-1, 0, 1\}$ };  $int my[3][3] = {$  $\{-1, -2, -1\},\$  $\{0, 0, 0\},\$  $\{1, 2, 1\}$ }; Sobel 算子, input 是输入文件名, output 是输出文 void PGMSobel2(char\* input, char\* 件名。支持 P5 格式的 PGM 图像。 XOutput 是输出的 XOutput, char\* X 方向的梯度图像, YOutput 是输出的 Y 方向的梯度 图像, SobelOutput 是输出的整幅图像的 Sobel 算子 YOutput, char\* SobelOutput, int 计算结果, min 和 max 是图像归一化的相关参数, 如 sobel x[3][3], int min=100, max=0. 参考模板: sobel\_y[3][3], int min, int max) int sobel  $x[3][3] = \{\{-1, 0, 1\}, \{-2, 0, 2\}, \{-1, 0, 1\}\}$  $0, 1\}\};$ int sobel  $v[3][3] = \{\{1, 2, 1\}, \{0, 0, 0\}, \{-1, -1\}\}$  $2, -1\}$ ; Sobel 算子, input 是输入文件名, output 是输出文 void Sobel (char\* input, char\* output) 件名。支持 PGM 文件。

void Laplatian(char*	Laplatian 算子, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 PGM 文件。
void HorizSobel(char*	水平 Sobel 算子,input 是输入文件名,output 是
input, char* output)	输出文件名。支持 P5 格式的 PGM 图像。
void VertSobel(char*	垂直 Sobel 算子,input 是输入文件名,output 是
input, char* output)	输出文件名。支持 P5 格式的 PGM 图像。
void PGMSobel1(char*	Sobel 算子, input 是输入文件名, output 是输出文
input, char*	件名。支持 P5 格式的 PGM 图像。threshold 是目标
output, int threshold)	阈值,如 threshold=80。
void	Sobel 算子, input 是输入文件名, output 是输出文
RAWSobelEdge(char*	件名。ROWS 是图像的行,COLS 是图像的列,M 是滤
input, char*	波相关参数,如 M=1。支持 RAW 图像。
output, int ROWS, int	参考模板:
COLS, int M, float	float sobelX[3][3] = {{-1,0,1},
sobelX[3][3], float	$\{-2, 0, 2\},$
	$\{-1, 0, 1\}\}$ :
sobe1Y[3][3])	{-1, 0, 1}};
	$floot coholy[0][0] = \{\{1, 0, 1\}\}$
	float sobelY[3][3] = $\{\{-1, -2, -1\},$
	{0, 0, 0},
	$\{1,2,1\}\};$
void	边缘检测,input 是输入文件名,output 是输出文
RAWPlaceHolder(char*	件名。ROWS 是图像的行,COLS 是图像的列,M 是滤
input, char*	波相关参数,如 M=1。支持 RAW 图像。
output, int ROWS, int	参考模板:
COLS, int M, float	float $mask[3][3] = \{\{-1, -2, -1\},\$
mask[3][3])	$\{0,0,0\},\{1,2,1\}\};$
void	拉普拉斯图像增强, input 是输入文件名, output 是
LaplacianEnhancement	输出文件名。如: N=1。支持 8 位 BMP 图像。
(char* input, char*	参考模板:
output, int N, int	int Lap1Mask[3][3] = {
Lap1Mask[3][3])	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void	拉普拉斯平滑, input 是输入文件名, output 是输
LaplaceSmooth(char*	出文件名。如: N=1。支持8位BMP图像。
input, char*	参考模板:
output, int N, int	int Lap1Mask[3][3] = {
Lap1Mask[3][3])	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void Sobell(char*	Sobel 算子, input 是输入文件名, output 是输出文
input, char*	件名。如: N=1。支持 8 位 BMP 图像。
Tubar, cuara	コンコットロ To My O LY DIM 日図。

```
参考模板:
output, int
              N, int
Sb1Mask1[3][3], int
                        int Sb1Mask1[3][3] = {
Sb1Mask2[3][3])
                                    -1, -2, -1,
                                     0, 0, 0,
                                     1, 2, 1
                        };
                        int Sb1Mask2[3][3] = {
                                    -1, 0, 1,
                                    -2, 0, 2,
                                    -1, 0, 1
                        }:
                     Sobel 平滑, input 是输入文件名, output 是输出文
void
SobelSmooth(char*
                     件名。如: N=1。支持 8 位 BMP 图像。
                     参考模板:
input, char*
                        int Sb1Mask1[3][3] = {
output, int
              N, int
Sb1Mask1[3][3], int
                                    -1, -2, -1,
Sb1Mask2[3][3])
                                     0, 0, 0,
                                     1, 2, 1
                        };
                        int Sb1Mask2[3][3] = {
                                    -1, 0, 1,
                                    -2, 0, 2,
                                    -1, 0, 1
                     Roberts 算子, input 是输入数据, output 是输出数
void Roberts (unsigned
char** input, unsigned
                     据。
char** output)
void Roberts (BMPMat**
                     Roberts 算子, input 是输入数据, output 是输出数
input, BMPMat**
                     据。
output)
                     Sobel 算子, 耗时较长, input 是输入文件名, output
void
                     是输出文件名。支持 24 位 BMP 图像。
SobelOperator (char*
input, char* output)
SobelImage**
                     返回处理后的各像素点的坐标和对应的像素值,若
SobelOperator (char*
                     是边缘点则对应白色,否则对应黑色。支持24位BMP
                     图像。
input)
                     需引入以下结构体:
                     typedef struct {
                         int x:
                         int y;
                         unsigned char red;
                         unsigned char green;
                         unsigned char blue;
                     }SobelImage;
```

<pre>void STLSection(char* input, char* output, int sliceAmount, int resolution, int c) void SURF(char* input1, char* input2, char* output)</pre>	STL 切片, input 是输入的 STL 文件, output 是输出的切片文件前缀名, sliceAmount 是切片量, 如: sliceAmount=50, resolution 是分辨率, 如: resolution=260, c是执行的相关参数, 如: c=5。  SURF 算子, input1 和 input2 是输入文件名, output 是输出文件名, 支持 BMP 图像。
<pre>void EdgeDetection(char* input, char* output)</pre>	边缘检测, input 是输入文件名, output 是输出文件名。支持 4 位 BMP 图像。
<pre>void EdgeDetection1(char* input, char* output, short sharpen[3][3])</pre>	边缘检测, input 是输入文件名, output 是输出文件名。支持 8 位 BMP 图像。 参考模板: short sharpen[3][3] = {{1, 1, 1}, {1, -8, 1}, {1, 1, 1}};
<pre>void EdgeDetection2(char* input, char* output, int a)</pre>	边缘检测,input 是输入文件名,output 是输出文件名。a 是用于设置图像像素的相关参数,如 a=3。 支持 24 位 BMP 图像。
<pre>void EdgeDetection3(char* input, char* output, int a)</pre>	边缘检测, input 是输入文件名, output 是输出文件名。a 是用于设置图像像素的相关参数,如 a=3。支持 24 位 BMP 图像。
void EdgeDetection4(char* input, char* output, int a)	边缘检测, input 是输入文件名, output 是输出文件名。a 是用于设置图像像素的相关参数,如 a=3。支持 24 位 BMP 图像。
<pre>void YFiltering(char* input, char* output, int sobel_x[3][3], int sobel_y[3][3])</pre>	Y 方向滤波, input 是输入文件名, output 是输出文件名。支持 P5 格式的 PGM 图像。 参考模板: int sobel_x[3][3] = { { 1, 0, -1}, { 2, 0, -2}, { 1, 0, -1}}; int sobel_y[3][3] = { { 1, 2, 1}, { 0, 0, 0}, {-1, -2, -1}};
<pre>void XFiltering(char* input, char* output, int sobel_x[3][3], int sobel_y[3][3])</pre>	X 方向滤波, input 是输入文件名, output 是输出文件名。支持 P5 格式的 PGM 图像。 参考模板: int sobel_x[3][3] = { { 1, 0, -1}, { 2, 0, -2}, { 1, 0, -1}};

	int sobel_y[3][3] = { { 1, 2, 1},
	$\{0, 0, 0\},$
	$\{-1, -2, -1\}\};$
void	Sobel 算子, input 是输入文件名, output 是输出文
SobelFiltering(char*	件名。支持 P5 格式的 PGM 图像。
input, char*	参考模板:
output, int	int sobel_x[3][3] = { { 1, 0, -1},
sobel_x[3][3], int	$\{2, 0, -2\},\$
sobe1_y[3][3])	$\{1, 0, -1\}\};$
	int sobel_y[3][3] = { { 1, 2, 1},
	$\{0, 0, 0\},\$
void	{-1, -2, -1}}; Prewitt 算子, input 是输入文件名, output 是输出
PrewittFiltering(cha	文件名。支持 P5 格式的 PGM 图像。
r* input, char*	参考模板:
output, int	int prewitt_ $x[3][3] = \{ \{ 5, 5, 5 \}, \}$
prewitt_x[3][3], int	$\{-3, 0, -3\},$
prewitt_y[3][3])	$\{ -3, -3, -3 \} \};$
	int prewitt_y[3][3] = { $\{5, -3, -3\},$
	$\{5, 0, -3\},\$
	{5, -3, -3}};
void	Laplace 算子, input 是输入文件名, output 是输出
LaplacianFiltering(c har* input, char*	文件名。支持 P5 格式的 PGM 图像。laplacian 是 Laplacian 算子模板。
output, int	参考模板:
laplacian[3][3])	int laplacian[3][3] = { { 1, 1, 1},
	{ 1, -8, 1},
	{ 1, 1, 1}};
void	Sobel 算子,支持 BMP 图像。
SobelOperation1(char	
* input, char*	
output, int width, int	
height)	Sobel 算子,支持 BMP 图像。
SobelOperation2(char	5000c1 开 1 , 文刊 DMI 国 图。
* input, char*	
output, int width, int	
height)	
void Roberts(char*	Roberts 边缘检测,支持 BMP 图像。
input, char* output)	
void Prewitt(char*	Prewitt 边缘检测,支持 BMP 图像。
input, char* output)	
void Sobel(char*	Sobel 边缘检测,支持 BMP 图像。
input, char* output)	

void Laplace(char*	Laplace 边缘检测,支持 BMP 图像。
input, char* output)	Notes A De Hilliam Co. In
void	高级方框模糊,参考: radius=5。支持 PNG 文件。
BoxBlurAdvanced(stri	
ng input, string	
output, int radius)	
void	霍夫变换, input 是输入的 RAW 文件, output 是输
HoughTransform(char*	出的 RAS 文件,threshold=100。
input, char*	
output, unsigned char	
threshold)	
static void	边缘检测,参考: a=0.33, b=0.33, c=0.33。支持24
EdgeDetectionWithout	位 BMP 图像。
Nonmaximum(const	
LPCTSTR input, const	
LPCTSTR output, double	
a, double b, double c)	
static void	边缘检测,参考: orank=20, oranb=80。支持 24 位
CannyEdgeDetection(c	BMP 图像。
onst LPCTSTR input,	
const LPCTSTR	
output, double	
a, double b, double	
c, int orank, int	
oranb)	
static void	霍夫变换,参考: a=0.33, b=0.33, c=0.33, orank=20,
HoughTransform(const	oranb=80。支持 24 位 BMP 图像。
LPCTSTR input, const	
LPCTSTR output, double	
a, double b, double	
c, int orank, int	
oranb)	
void	基础方框模糊,支持 PNG 文件。
BoxBlurBasic(string	
input, string output)	
void	Sobel 算子, input 是输入文件名, output 是输出文
SobelSharpen(char*	件名。Templatex 是 laplace 锐化模板,4 邻域,
input, char*	Templatey 是 laplace 锐化模板,8 邻域,
output, int	coefficient1 = 9, coefficient2 = 16。支持24位
Templatex[3][3], int	BMP 图像。
Templatey[3][3], int	
coefficient1, int	
coefficient2)	
void	图像加解密,Key 是密钥,a=1 时执行加密,a=0 时

FranchionDogramtion	执行解密。支持 24 位 BMP 图像。
EncryptionDecryption	1)八八肝品。又行 24 位 DMF 图像。
(char* input, char*	
output, int Key, int a)	图塔加索 · ,日处)之体力 , ,日处山之
void Encryption(char*	图像加密,input 是输入文件名,output 是输出文
input, char*	件名。Key 是密钥。支持 24 位 BMP 图像。
output, int Key)	
void Decryption(char*	图像解密,input 是输入文件名,output 是输出文
input, char*	件名。Key 是密钥。支持 24 位 BMP 图像。
output, int Key)	
void Nesting(char*	图像嵌套,Biginput 是输入的大图,Smallinput 是
Biginput, char*	输入的小图。支持 24 位 BMP 图像。
Smallinput, char*	
output)	
void Blend(char*	图像融合之混合化, input1 和 input2 是输入的两个
input1, char*	要融合的图像, output 是输出文件名。支持 24 位
input2, char* output)	BMP 图像。
void Checker (char*	图像融合之棋盘化, input1 和 input2 是输入的两个
input1, char*	要融合的图像, output 是输出文件名。支持 24 位
input2, char* output)	BMP 图像。
void Blend1 (char*	图像融合之混合化, input1 和 input2 是输入的两个
input1, char*	要融合的图像,output 是输出文件名。支持 24 位
input2, char* output)	BMP 图像。
void Checker1 (char*	图像融合之棋盘化, input1 和 input2 是输入的两个
input1, char*	要融合的图像,output 是输出文件名。支持 24 位
input2, char* output)	BMP 图像。
void	二维码生成,filename 是生成的二维码图像文件名,
QRCodeGeneration(cha	inputString 是二维码包含的信息。支持 BMP 图像。
r *filename, char*	Tilputotiffig 在一维问色自用自恋。文诗 Dini 图像。
<pre>inputString) vector<float></float></pre>	角点检测,支持 P5 和 P6 格式的 PPM 文件,从返回
	用点检测,文持 P5 和 P6 格式的 PPM 文件,从返回     值数组中的第一个元素开始,返回值以 3 个元素为
HarrisCornerDetectio	
n(char* input, int	一组,分别是角点的 X 坐标、Y 坐标和分数,若返回
width, int height, int	值数组名为 A, 则 {A[0], A[1], A[2]} 是第一个角点的
channels, int	数据, {A[3], A[4], A[5]} 是第二个角点的数据,以此
step, float	类推。
threshold, float	input 是输入的图像文件名,width 和 height 是输
k, float sigma)	入图像的宽和高,channels 是输入图像的通道数,
	step 默认=-1, threshold 是 Harris 检测中角点的
	得分阈值,k是 Harris 评分函数中的 k值,sigma 是
	用于 IxIy 阵列平滑的 sigma 值,参考:
	threshold=2000, k=1, sigma=1.2.
vector <float></float>	角点检测,支持 P5 和 P6 格式的 PPM 文件,从返回
HarrisCorner(char*	值数组中的第一个元素开始,返回值以 3 个元素为
input, char*	一组,分别是角点的X坐标、Y坐标和分数,若返回

output, int width, int height, int channels, float threshold, float k, float sigma) 值数组名为 A,则  $\{A[0], A[1], A[2]\}$  是第一个角点的数据, $\{A[3], A[4], A[5]\}$  是第二个角点的数据,以此类推。

input 是输入的图像文件名, width 和 height 是输入图像的宽和高, channels 是输入图像的通道数, threshold 是 Harris 检测中角点的得分阈值, k 是 Harris 评分函数中的 k 值, sigma 是用于 IxIy 阵列 平滑的 sigma 值, 参考: threshold=2000, k=1, sigma=1.2。

int\*

TemplateMatching (cha
r\* input1, char\*
input2, char\*
output, unsigned char
red, unsigned char
green, unsigned char
blue, double
MatchScore)

模板匹配,返回值中第一个元素是匹配框左上角的 纵坐标,第二个元素是匹配框左下角的纵坐标,第三 个元素是匹配框的左上角的横坐标,第四个元素是 匹配框的右上角的横坐标。input1 是搜索图像, input2 是模板图像,output 是匹配结果图像, MatchScore=0.9。支持BMP图像。

int\*

TemplateMatching (cha
r\* input1, char\*
input2, unsigned char
red, unsigned char
green, unsigned char
blue, double
MatchScore)

模板匹配,返回值中第一个元素是匹配框左上角的 纵坐标,第二个元素是匹配框左下角的纵坐标,第三 个元素是匹配框的左上角的横坐标,第四个元素是 匹配框的右上角的横坐标。input1 是搜索图像, input2 是模板图像,output 是匹配结果图像, MatchScore=0.9。支持BMP图像。

struct imagine TemplateMatching(str uct imagine ColorSource, struct imagine input1, struct imagine input2, unsigned char red, unsigned char green, unsigned char blue, double MatchScore)

模板匹配,返回匹配结果的图像,input1 是搜索图像,input2 是模板图像,MatchScore=0.9。支持BMP图像。

需引入以下结构体:

typedef struct imagine {
 unsigned char \*R,\*G,\*B,\*header;
 int W, H, Wpad, size;
};

声明:

void grayscale\_image(char\*

nume\_fisier\_sursa, char\*
nume\_fisier\_destinatie);

struct imagine salvareBitmap (char
\*destinatieFisier);

void afisare (struct imagine img, char \*destinatieSalvare); 将灰度应用于所有图像:

struct imagine output;

```
grayscale_image(inputImage1,
                     input1 grayscale);
                         grayscale image(inputImage2,
                     input2 grayscale);
                         ColorSource=salvareBitmap(inputImage1);
                     //inputImagel 是彩色原图
                         input1=salvareBitmap(input1 grayscale);
                         input2=salvareBitmap(input2 grayscale);
                     output=TemplateMatching(ColorSource, input1, i
                     nput2, red, green, blue, MatchScore);
                         afisare (output, outputfile);
                     模板匹配,返回值中第一个元素是匹配框左上角的
int*
                     纵坐标,第二个元素是匹配框左下角的纵坐标,第三
TemplateMatching(str
                     个元素是匹配框的左上角的横坐标,第四个元素是
             imagine
input1, struct imagine
                     匹配框的右上角的横坐标。input1 是搜索图像,
                     input2 是模板图像, output 是匹配结果图像,
input2, unsigned
               char
                     MatchScore=0.9。支持 BMP 图像。
red, unsigned
               char
green, unsigned
                     需引入以下结构体:
               char
blue, double
                     typedef struct imagine {
MatchScore)
                         unsigned char *R, *G, *B, *header;
                         int W, H, Wpad, size;
                     };
                     声明:
                     void
                                           grayscale image(char*
                     nume_fisier_sursa, char*
                     nume fisier destinatie);
                     struct
                               imagine
                                          salvareBitmap
                                                           (char
                     *destinatieFisier);
                     将灰度应用于所有图像:
                         struct imagine input1, input2;
                         grayscale image(inputImage1,
                     input1 grayscale);
                         grayscale_image(inputImage2,
                     input2 grayscale);
                         input1=salvareBitmap(input1_grayscale);
                         input2=salvareBitmap(input2 grayscale);
void
                     模板匹配,suprapunereMaxima 表示最大重叠率,参
                                 MaximumMatchingQuantity=10
TemplateMatching(cha
                     考
         input, char*
                     MatchScore=0.8, suprapunereMaxima=0.2。支持 BMP
templatename, char*
                     图像。
output, unsigned
MaximumMatchingQuant
ity, double
```

MatchScore, float suprapunereMaxima, un signed char red, unsigned char green, unsigned char blue) int* TemplateMatching(cha	模板匹配,返回值是匹配框的左上角坐标(x,y)。 支持 BMP 图像。
r* input1, char* input2, char* output) int*	模板匹配,返回值是匹配框的左上角坐标(x,y)。
TemplateMatching1(ch ar* input1, char* input2)	支持 BMP 图像。
<pre>int* TemplateMatching2(ch ar* input1, char* input2)</pre>	模板匹配,返回值是匹配框的左上角坐标(x,y)。 支持 BMP 图像。
my_image_comp* TemplateMatching(my_ image_comp input, my_image_comp Template, int H, int length, float* hpf)	模板匹配,返回匹配结果。支持BMP图像。 需引入以下结构体: struct my_image_comp { int width; int border; int border; float *handle; float *buf; my_image_comp() { width = height = stride = border = 0; handle = buf = NULL; } ~my_image_comp() { if (handle != NULL) delete[] handle; } void init(int height, int width, int border) { this->width = width; this->height = height; this->border = border; stride = width + 2*border; if (handle != NULL) delete[] handle; handle = new
	<pre>float[stride*(height+2*border)];      buf = handle + (border*stride) + border;</pre>

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

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

```
{ width = height = stride = border = 0;
                      handle = buf = NULL; }
                          my image comp()
                            { if (handle != NULL) delete[] handle; }
                         void init(int height, int width, int
                      border)
                              this->width = width; this->height =
                      height; this->border = border;
                              stride = width + 2*border;
                              if (handle != NULL)
                                delete∏ handle:
                              handle
                                                               new
                      float[stride*(height+2*border)];
                             buf = handle + (border*stride) +
                      border:
                         void perform boundary extension();
                      };
                      struct filt {
                         float* centre:
                         int length;
                      }:
                      声明:
                      int
                          read bmp(char*
                                           image,
                                                    my image comp*
                      input_comps, int* num_comps, int H);
                      int write_bmp(my_image_comp* output_comps,
                      char* dest):
                      filt make filter(int type);
                      参考:
                      my_image_comp input;
                      my image comp Template;
                      filt filter = make filter(1);
                      int length = filter.length;
                      float* hpf = filter.centre;
                      int H = (filter. length - 1) / 2;
                      int num comps = 1;
                      read_bmp(inputfile, &input, &num_comps, 0);
                      read bmp (Templatefile, &Template, &num comps,
                      H);
                      write bmp(&input, outputfile);
                      模板匹配,返回值是匹配框的左上角坐标(x,v)。
int*
TemplateMatching(cha
                      min 是与匹配分数相关的参数,支持 BMP 图像。参
                     考: min=65026。
        input1, char*
```

input2, char*	
output, float min)	
int*	模板匹配,返回值是匹配框的左上角坐标(x,y)。
TemplateMatching(bmp	min 是与匹配分数相关的参数,支持 BMP 图像。参
_in input1, bmp_in	考: min=65026。
input2, float min)	需引入以下结构体:
inputz, iioat min)	struct bmp_in {
	int num_components, rows, cols;
	int num_components, rows, cors,
	int line_bytes;
	int alignment_bytes;
	FILE *in;
	<b>}</b> ;
	声明:
	extern int bmp_in_open(bmp_in *state, const
	char *fname);
	参考:
	bmp_in input1, input2;
	<pre>bmp_in_open(&amp;input1, input1file);</pre>
	bmp_in_open(&input2, input2file);
double*	模板匹配,支持 JPG 图像,返回值的第 1 和第 2 个
TemplateMatch(byte**	元素是匹配框左上角顶点的横坐标和纵坐标,第3个
* input, byte***	元素是目标与模板相对的旋转角度,第 4 个元素是
Template, char*	缩放比例。参考: size=1, best_loss=1000000000,
output, int irows, int	a=0.5, b=2.1, c=0.5, d=45, e1=20, e2=20.
icols, int trows, int	声明:
tcols, int size, int	#define byte unsigned char
best_loss, double	byte ***LoadRgb(const char *fname, int *rows,
a, double b, double	int *cols, int *chan);
c, double d, int el, int	参考:
e2)	int irows, icols, ichan;
	int trows, tcols, tchan;
	byte*** input = LoadRgb(inputfile, &irows,
	&icols, &ichan);
	byte*** Template = LoadRgb(templatefile,
	&trows, &tcols, &tchan);
double*	模板匹配,支持 JPG 图像,返回值的第1和第2个
TemplateMatch(byte**	元素是匹配框左上角顶点的横坐标和纵坐标,第3个
* input, byte***	元素是目标与模板相对的旋转角度,第 4 个元素是
Template, int	缩放比例。参考: size=1, best_loss=1000000000,
irows, int icols, int	a=0.5, b=2.1, c=0.5, d=45, e1=20, e2=20.
trows, int tcols, int	声明:
size, int	#define byte unsigned char
best_loss, double	byte ***LoadRgb(const char *fname, int *rows,

	,
a, double b, double	
c, double d, int el, int	参考:
e2)	int irows, icols, ichan;
	int trows, tcols, tchan;
	byte*** input = LoadRgb(inputfile, &irows,
	&icols, &ichan);
	byte*** Template = LoadRgb(templatefile,
	&trows, &tcols, &tchan);
byte***	模板匹配,支持 JPG 图像,返回匹配结果。参考:
TemplateMatch1(byte*	size=1, best_loss=1000000000, a=0.5, b=2.1,
** input, byte***	c=0.5, d=45, e1=20, e2=20.
Template, int	声明:
irows, int icols, int	#define byte unsigned char
trows, int tcols, int	byte ***LoadRgb(const char *fname, int *rows,
size, int	int *cols, int *chan);
best_loss, double	void SaveRgbPng(byte ***in, const char
a, double b, double	*fname, int rows, int cols);
	参考:
c, double d, int el, int	
e2)	int irows, icols, ichan;
	int trows, tcols, tchan;
	byte*** input = LoadRgb(inputfile, &irows,
	&icols, &ichan);
	byte*** Template = LoadRgb(templatefile,
	&trows, &tcols, &tchan);
int	模板匹配,返回匹配到的目标数量。支持 BMP 图像。
ObjectFind(bmpread_t	需引入以下头文件:
input, bmpread_t	#include "bmpread1.h"
Template)	参考:
_	<pre>bmpread_t input, Template;</pre>
	bmpread(inputfile, BMPREAD_BYTE_ALIGN
	BMPREAD_ANY_SIZE, &input);
	bmpread(Templatefile, BMPREAD BYTE ALIGN
	BMPREAD_ANY_SIZE, &Template);
double*	模板匹配,返回值数组: 匹配框左上角顶点的 X 和 Y
	操似匹配,返回值数组: 匹配框左上用项点的 A 和 I     坐标、模板的宽和高、差异度。参考: output 是匹
TemplateMatching1(Im	
age2* input, Image2*	配结果图像名,output_txt 是保存的匹配相关数据
Template, char*	的文本文件, threshold=0.5 ,
output, char*	isWriteImageResult=1,color是当图像是灰度图时
output_txt, double	的匹配框颜色,red、green 和 blue 是当图像是彩色
threshold, int	图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。
isWriteImageResult,u	需要引入以下结构体:
nsigned char	typedef struct Image2
color, unsigned char	{
red, unsigned char	int width;

```
green, unsigned
               char
                       int height;
blue)
                       int channel;
                       unsigned char* data;
                    } Image2;
                    声明:
                    Image2* readPXM(const char* name);
                    Image2* input = readPXM(inputFileName);
                    Image2* Template = readPXM(templatename);
double*
                    模板匹配, 返回值数组: 匹配框左上角顶点的 X 和 Y
                    坐标、模板的宽和高、差异度。参考: output txt 是
TemplateMatching1(Im
age2* input, Image2*
                    保存的匹配相关数据的文本文件, threshold=0.5,
                    isWriteImageResult=1, color 是当图像是灰度图时
Template, char*
                    的匹配框颜色, red、green 和 blue 是当图像是彩色
output txt, double
                    图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。
threshold, unsigned
char
      color, unsigned
                    需要引入以下结构体:
                    typedef struct Image2
char
        red, unsigned
char
      green, unsigned
char blue)
                       int width:
                       int height;
                       int channel;
                       unsigned char* data;
                    } Image2;
                    声明:
                    Image2* readPXM(const char* name);
                    参考:
                    Image2* input = readPXM(inputFileName);
                    Image2* Template = readPXM(templatename);
                    模板匹配,返回值数组: 匹配框左上角顶点的 X 和 Y
double*
TemplateMatching2(Im
                    坐标、模板的宽和高、差异度。参考: output 是匹
age2* input,
                    配结果图像名, output txt 是保存的匹配相关数据
            Image2*
Template, char*
                    的
                        文
                            本
                                 文
                                             threshold=0.5
                    isWriteImageResult=1,color 是当图像是灰度图时
output, char*
                    的匹配框颜色, red、green 和 blue 是当图像是彩色
output txt, double
                    图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。
threshold, int
                    需要引入以下结构体:
isWriteImageResult, u
                    typedef struct Image2
nsigned
               char
color, unsigned
               char
                       int width:
red, unsigned
               char
green, unsigned
               char
                       int height;
blue)
                       int channel;
                       unsigned char* data;
                    }Image2;
                    声明:
```

```
Image2* readPXM(const char* name);
                    参考:
                    Image2* input = readPXM(inputFileName);
                    Image2* Template = readPXM(templatename);
double*
                    模板匹配,返回值数组: 匹配框左上角顶点的 X 和 Y
                    坐标、模板的宽和高、差异度。参考: output txt 是
TemplateMatching2(Im
                    保存的匹配相关数据的文本文件, threshold=0.5,
age2* input, Image2*
                    isWriteImageResult=1,color 是当图像是灰度图时
Template, char*
                    的匹配框颜色, red、green 和 blue 是当图像是彩色
output txt, double
threshold, unsigned
                    图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。
     color, unsigned
                    需要引入以下结构体:
char
                    typedef struct Image2
char
       red, unsigned
      green, unsigned
char
char blue)
                       int width;
                       int height;
                       int channel;
                       unsigned char* data;
                    } Image2;
                    声明:
                    Image2* readPXM(const char* name);
                    Image2* input = readPXM(inputFileName);
                    Image2* Template = readPXM(templatename);
Image2*
                    模板匹配,返回匹配结果的图像数据。参考:
TemplateMatching3(Im
                    output txt 是保存的匹配相关数据的文本文件,
                    threshold=0.5, isWriteImageResult=1, color 是
age2* input,
            Image2*
                    当图像是灰度图时的匹配框颜色, red、green和blue
Template,
              char*
output_txt,
             double
                    是当图像是彩色图时的匹配框颜色的红绿蓝通道
threshold,
               int
                    值。
                    需要引入以下结构体:
isWriteImageResult,
unsigned char color,
                    typedef struct Image2
unsigned
         char red.
unsigned char green,
                       int width:
unsigned char blue)
                       int height;
                       int channel;
                       unsigned char* data;
                    } Image2;
                    声明:
                    Image2* readPXM(const char* name);
                    参考:
                    Image2* input = readPXM(inputFileName);
                    Image2* Template = readPXM(templatename);
                    图像匹配,返回匹配结果的图像数据。参考:
Image2*
                    output_txt 是保存的匹配相关数据的文本文件,
TemplateMatching4(Im
```

```
age2* input,
                     threshold=0.5, isWriteImageResult=1, color 是
             Image2*
                     当图像是灰度图时的匹配框颜色, red、green和blue
Template,
               char*
                     是当图像是彩色图时的匹配框颜色的红绿蓝通道
output txt,
              double
threshold,
                 int
                     信。
isWriteImageResult,
                     需要引入以下结构体:
unsigned char color,
                     typedef struct Image2
          char red,
unsigned
unsigned char green,
                         int width;
unsigned char blue)
                         int height;
                         int channel;
                         unsigned char* data;
                     }Image2;
                     声明:
                     Image2* readPXM(const char* name);
                     参考:
                     Image2* input = readPXM(inputFileName);
                     Image2* Template = readPXM(templatename);
double*
                     模板匹配,返回匹配框的中心点坐标、旋转角度和缩
                     放比例。参考: c=0.5, threshold=0.9。 支持 PPM 文
TemplateMatching (RGB
PACKED IMAGE* input,
                     件。
                     声明:
RGB PACKED IMAGE*
Template, char*
                     #ifndef P
output, unsigned char
                     #if defined( STDC ) | defined( cplusplus)
                     #define P(protos) protos
red, unsigned
                char
                     #else
green, unsigned
                char
                     #define __P(protos) ()
blue, double c, double
threshold)
                     #endif
                     #endif
                     RGB PACKED IMAGE
                                              *readRGBPackedImage
                      P(( char* )):
                     需引入以下结构体:
                     typedef struct rgb packed pixel {
                         BYTE r;
                         BYTE g;
                         BYTE b:
                     } RGB PACKED PIXEL;
                     typedef struct rgb packed image {
                         int cols;
                         int rows;
                         RGB PACKED PIXEL **p;
                         RGB PACKED PIXEL *data p;
                     } RGB PACKED IMAGE;
                     参考:
                     RGB PACKED IMAGE*
                                              Template
```

```
readRGBPackedImage(templatename);
                      RGB PACKED IMAGE*
                                                 input
                      readRGBPackedImage(inputFileName);
                      模板匹配, 返回匹配框的中心点坐标、旋转角度和缩
double*
TemplateMatching1(RG
                      放比例。参考: c=0.5, threshold=0.9。 支持 PPM 文
B PACKED IMAGE*
                      件。
                      声明:
input,
RGB PACKED IMAGE*
                      #ifndef P
                      #if defined( STDC ) || defined( cplusplus)
Template, unsigned
        red, unsigned
                      #define __P(protos) protos
char
char
      green, unsigned
                      #else
                      #define __P(protos) ()
         blue, double
char
c, double threshold)
                      #endif
                      #endif
                      RGB PACKED IMAGE
                                               *readRGBPackedImage
                      P(( char* ));
                      需引入以下结构体:
                      typedef struct rgb packed pixel {
                         BYTE r:
                         BYTE g;
                         BYTE b;
                      } RGB PACKED PIXEL;
                      typedef struct rgb packed image {
                          int cols;
                          int rows;
                         RGB PACKED PIXEL **p;
                         RGB PACKED PIXEL *data p;
                      } RGB PACKED IMAGE;
                      参考:
                      RGB PACKED IMAGE*
                                                Template
                      readRGBPackedImage(templatename);
                      RGB PACKED IMAGE*
                                                 input
                      readRGBPackedImage(inputFileName);
                      模板匹配,返回匹配框的中心点坐标、旋转角度和缩
double*
TemplateMatching2(RG
                      放比例。参考: c=0.5, threshold=0.9。支持 PPM 文
B PACKED IMAGE*
                      件。
                      声明:
input,
                      #ifndef P
RGB PACKED IMAGE*
                      #if defined( STDC ) || defined( cplusplus)
Template, unsigned
char
        red, unsigned
                      #define __P(protos) protos
char
      green, unsigned
                      #else
                      #define P(protos) ()
         blue, double
char
c, double threshold)
                      #endif
                      #endif
```

```
RGB PACKED IMAGE
                                               *readRGBPackedImage
                       P(( char* ));
                      需引入以下结构体:
                      typedef struct rgb packed pixel {
                          BYTE r:
                          BYTE g;
                          BYTE b;
                      } RGB PACKED PIXEL;
                      typedef struct rgb packed image {
                          int cols;
                          int rows;
                          RGB_PACKED_PIXEL **p;
                          RGB PACKED PIXEL *data p;
                      } RGB PACKED IMAGE;
                      参考:
                      RGB PACKED IMAGE*
                                                Template
                      readRGBPackedImage(templatename);
                      RGB PACKED IMAGE*
                      readRGBPackedImage(inputFileName);
RGB_PACKED IMAGE*
                      模板匹配,返回匹配结果的图像数据。参考: c=0.5,
                      threshold=0.9。支持 PPM 文件。
TemplateMatching (RGB
                      声明:
PACKED IMAGE* input,
RGB PACKED IMAGE*
                      #ifndef P
                      #if defined(__STDC__) || defined(__cplusplus)
Template, unsigned
                      #define P(protos) protos
char
     red.
            unsigned
char green, unsigned
                      #else
char blue, double c,
                      #define P(protos) ()
double threshold)
                      #endif
                      #endif
                      RGB PACKED IMAGE
                                               *readRGBPackedImage
                      P(( char* ));
                      需引入以下结构体:
                      typedef struct rgb packed pixel {
                          BYTE r;
                          BYTE g;
                          BYTE b;
                      } RGB PACKED PIXEL;
                      typedef struct rgb packed image {
                          int cols;
                          int rows;
                          RGB PACKED PIXEL **p;
                          RGB PACKED PIXEL *data p;
                      } RGB PACKED IMAGE;
                      参考:
```

	DOD DACKED THACE.			
	RGB_PACKED_IMAGE* Template =			
	readRGBPackedImage(templatename);			
	RGB_PACKED_IMAGE* input =			
	readRGBPackedImage(inputFileName);			
int*	模板匹配,支持 24 位 BMP 图像,返回匹配框的左上			
TemplateMatching(cha	角坐标(x, y)。			
r* input, char*				
Template)				
int*	模板匹配,支持 24 位 BMP 图像,返回匹配框的左上			
TemplateMatching(bmp	角坐标 (x, y)。			
_img input,bmp_img	需引入以下结构体和声明:			
Template)	enum bmp_error			
,	{			
	BMP_FILE_NOT_OPENED = -4,			
	BMP HEADER NOT INITIALIZED,			
	BMP INVALID FILE,			
	BMP ERROR,			
	BMP OK = 0			
	_			
	};			
	typedef struct _bmp_img			
	{			
	<pre>bmp_header img_header;</pre>			
	<pre>bmp_pixel **img_pixels;</pre>			
	} bmp_img;			
	声明:			
	<pre>enum bmp_error bmp_img_read(bmp_img *, const</pre>			
	char *);			
	参考例程:			
	<pre>bmp_img input, Template;</pre>			
	<pre>bmp_img_read(&amp;input, inputfile);</pre>			
	<pre>bmp_img_read(&amp;Template, outputfile);</pre>			
int	特征检测,返回找到的特征点的数量。a是亮度值,			
FeatureDetection(cha	如 a=255, b 与图像清晰度有关,默认 b=100; red、			
r* input, char*	green、blue 分别是特征点指示器颜色的红色、绿色、			
output, float a, int	蓝色的通道值,如 red=0, green=0, blue=1。支持			
b, float red, float	PPM 文件。			
green, float blue)				
unsigned int	特征检测,返回特征点数量。支持 PPM 文件。			
FeatureDetection(cha	14 声音04/ ~ 日 14 正 1 2 2 2 2 1 1 1 1 1 2 1 2			
r* input, char*				
output, unsigned char				
red, unsigned char				
_				
green, unsigned char				
blue)				

vector<int>
FeatureMatching(char
\* input1, char\*
input2, char\*
output, unsigned char
red, unsigned char
green, unsigned char
blue)

特征匹配,返回值中的数据格式为:第1个值是特征对的序号,从0开始,第2个值和第3个值分别是是图像1的其中一个特征点的横坐标和纵坐标,第4个值和第5个值分别是是图像2的其中一个特征点的横坐标和纵坐标并与第2个值和第3个值所在的图像1的特征点相对应,这5个值构成一组;同理,第6个值就是下一个特征对的序号,即为1,后面的值则以此类推。格式如:特征对:%d,图像1(%d,%d)->图像2:(%d,%d)。支持PPM文件。

int
FeatureMatching(char
\* input1, char\*
input2, char\* output,

input2, char\* output, float a, int b, float red, float green, float blue) 特征匹配,返回找到的特征匹配点的数量。a 是亮度值,如 a=255,b 与图像清晰度有关,默认 b=100;red、green、blue 分别是匹配线颜色的红色、绿色、蓝色的通道值,如 red=0, green=0,blue=1。支持PPM文件。

unsigned int
FeatureMatching1(cha
r\* input1, char\*
input2, char\*
output, unsigned char
red, unsigned char
green, unsigned char
blue, bool
bExtractDescriptor)

从两幅输入图像中检测特征点,然后使用蛮力方法 匹配两幅图像的特征。input 是输入图像,output 是 生成的特征点图像,bExtractDescriptor=true。返 回匹配的特征点数量。支持 PGM 文件。

vector<int>
FeatureMatching2(cha
r\* input1, char\*
input2, char\*
output, unsigned char
red, unsigned char
green, unsigned char
blue, bool
bExtractDescriptor)

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

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

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

```
返回特征点列表。input 是输入图像, output 是生
std::list<ezsift::Si
ftKeypoint>
                    成的特征点图像,bExtractDescriptor=true。支持
FeatureExtraction2(c
                    PGM 文件。
                    需引入以下命名空间:
har*
        input, char*
output, unsigned char
                    namespace ezsift {
red, unsigned
                    #define DEGREE (128)
                                         //SIFT 关键点: 128 维
               char
green, unsigned
               char
                    struct SiftKeypoint {
blue, bool
                        int octave; // octave 数量
                                   // layer 数量
bExtractDescriptor)
                        int layer;
                        float rlayer; // layer 实际数量
                                  // 归一化的 row 坐标
                        float r:
                        float c:
                                   // 归一化的 col 坐标
                        float scale; // 归一化的 scale
                                         // row 坐标(layer)
                        float ri:
                        float ci;
                                         // column 坐标(layer)
                        float layer scale; // scale(layer)
                        float ori; // 方向(degrees)
                        float mag; // 模值
                        float descriptors[DEGREE]; //描述符
                    };
                    查找缺陷位置,返回材料缺陷的位置,以每4个元
vector<int>
DefectLocation (PGMDa
                    素为一组。Template 是模板图像, Sample 是样本图
ta* Template, PGMData*
                    像,g是缺陷的有效界限的 X 轴长度,h 是缺陷的有
Sample, int floor, int
                    效界限的 Y 轴长度,参考: floor=80, size=10,
                    a=64, b=64, c=16, d=16, e=2, f=4, g=65, h=65,
size, int a, int b, int
                    FULL=0, EMPTY=255, report=true.
c, int
       d, int
              e, int
                    引入以下结构体:
f, int
       g, int
              h, int
FULL, int
         EMPTY, boo1
                    typedef struct _PGMData {
report)
                        int row:
                        int col;
                        int max_gray;
                        int **matrix;
                    } PGMData;
                    若模板文件名为 Template, 样本文件名为 Sample,
                    使用以下代码获得合适的输入数据:
                    首先声明 readPGM 函数:
                    PGMData* readPGM(const
                                                   *file name,
                                            char
                    PGMData *data):
                    然后执行以下代码:
                    PGMData*
                                         model
```

(PGMData\*) malloc(sizeof(PGMData)):

data

readPGM(Template, model);

PGMData\*

```
(PGMData*) malloc(sizeof(PGMData));
                    readPGM(Sample, data);
                    之后将 model 和 data 传入相应的函数。
                    支持 P5 格式的 PGM 文件。
                    缺陷尺寸,返回缺陷尺寸,以4个为一组。Template
vector<int>
DefectSize(PGMData*
                    是模板图像, Sample 是样本图像, g 是缺陷的有效
                    界限的 X 轴长度, h 是缺陷的有效界限的 Y 轴长度,
Template, PGMData*
Sample, int floor, int
                    参考: floor=80, size=10, a=64, b=64, c=16, d=16,
                    e=2, f=4, g=65, h=65, FULL=0, EMPTY=255,
size, int a, int b, int
       d, int
                    report=true.
c, int
              e, int
f, int
       g, int
              h, int
                    引入以下结构体:
FULL, int
          EMPTY, bool
                    typedef struct PGMData {
report)
                        int row;
                        int col;
                        int max gray;
                        int **matrix;
                    }PGMData;
                    若模板文件名为 Template, 样本文件名为 Sample,
                    使用以下代码获得合适的输入数据:
                    首先声明 readPGM 函数:
                    PGMData*
                             readPGM(const
                                                   *file name,
                                             char
                    PGMData *data);
                    然后执行以下代码:
                    PGMData*
                                          mode1
                    (PGMData*) malloc(sizeof(PGMData));
                    readPGM(Template, model);
                    PGMData*
                                          data
                    (PGMData*) malloc(sizeof(PGMData));
                    readPGM(Sample, data);
                    之后将 model 和 data 传入相应的函数。
                    支持 P5 格式的 PGM 文件。
vector<int>
                    样品好坏的数量,返回结果中,第一个元素是合格的
                    圆圈数量,第二个元素是缺陷的圆圈数量。Template
GoodBadQuantity (PGMD
                    是模板图像, Sample 是样本图像, g 是缺陷的有效
ata*
Template, PGMData*
                    界限的 X 轴长度, h 是缺陷的有效界限的 Y 轴长度,
Sample, int floor, int
                    参考: floor=80, size=10, a=64, b=64, c=16, d=16,
size, int a, int b, int
                    e=2, f=4, g=65, h=65, FULL=0, EMPTY=255,
c, int
                    report=true.
       d, int
              e, int
                    引入以下结构体:
f, int
       g, int
              h, int
FULL, int
          EMPTY, bool
                    typedef struct _PGMData {
report)
                        int row;
                        int col;
                        int max_gray;
                        int **matrix;
```

	}PGMData; 若模板文件名为 Template, 样本文件名为 Sample, 使用以下代码获得合适的输入数据: 首先声明 readPGM 函数: PGMData* readPGM(const char *file_name, PGMData *data); 然后执行以下代码: PGMData* model = (PGMData*) malloc(sizeof(PGMData)); readPGM(Template, model);		
	PGMData* data = (PGMData*) malloc(sizeof(PGMData));		
	readPGM(Sample, data); 之后将 model 和 data 传入相应的函数。		
- 1 - 1	支持 P5 格式的 PGM 文件。		
struct hough_param_circle*	圆检测,返回找到的圆的位置和大小等相关信息。支持 RAW 文件。		
CircleDetection(char	需引入以下结构体:		
* input, int width, int	struct hough_param_circle {		
height)	int a;		
	int b;		
	int radius;		
	int resolution;		
	int thresh;		
	struct point *points;		
	<pre>int points_size;</pre>		
. 1			
unsigned int*	圆检测,返回圆心的坐标和半径,返回值数组中第一个元素是圆心 X 坐标,第二个元素是圆心 Y 坐标,		
CircleDetection(char * input)	第三个元素是圆半径。支持 BMP 图像。		
int Equal(char*	弗二个兀系是國丰任。文持 BMP 图像。   若比对图像的梯度幅相似性偏差值等于 c 则通过。		
input1, char*	input1 和 input2 是要比对的两个图像。c 是参考的		
input2, double c)	input 和 input 是安比约的两个图像。C 是多考的   阈值。支持 24 位 BMP 图像。		
int GreaterThan(char*	若比对图像的梯度幅相似性偏差值大于 c 则通过。		
input1, char*	input1 和 input2 是要比对的两个图像。c 是参考的		
input2, double c)	阈值。支持 24 位 BMP 图像。		
int LessThan(char*	若比对图像的梯度幅相似性偏差值小于 c 则通过。		
input1, char*	input1 和 input2 是要比对的两个图像。c 是参考的		
input2, double c)	阈值。支持 24 位 BMP 图像。		
double GMSD(char*	求两幅图像的梯度幅相似性偏差值并返回结果。		
input1, char* input2)	input1 和 input2 是要比对的两个图像。支持 24 位     BMP 图像。		
vector <vector<double< td=""><td>图像标定,返回各个顶角的坐标。参考: A_height =</td></vector<double<>	图像标定,返回各个顶角的坐标。参考: A_height =		
>> Correction(string	297, A_width = 210, thre_val=0.5, sigma=6.0,		

input, char* output,	threshold=3.5, angle_num=180, range_thre=5,
int A_height, int	fun_thre=0.8,point_thre=3。支持 BMP 文件。
A_width, double	
thre_val, double	
sigma, double	
threshold, int	
angle_num, int	
range_thre, double	
fun_thre, int	
point_thre)	
vector <double></double>	图像标定,返回各个直线的交点的 X 轴和 Y 轴坐标,
CalibrationAndCorrec	若以 vector <double> p 接收返回值则交点 1 的坐标  </double>
tion(char*	为(p[0], p[1]),交点 2 的坐标为(p[2], p[3]),以
input, char*	此类推。参考: sigma=5.5, gra_threshold=30,
output1, char*	vote_threshold=1000 , peak_dis=200 ,
output2, double	lines_color[3]= $\{0, 255, 0\}$
sigma, double	intersections_color[3]={0,0,255}。至少支持BMP
gra_threshold, double	图像。
vote_threshold,	
double	
peak_dis, unsigned	
char* lines_color,	
unsigned char*	
intersections_color)	
float*	图像匹配,返回值中的前 10 个元素是按顺序对应的
ImageMatching(char*	目标与模板的差异分数,最后一个元素是匹配到的
TargetImage, char*	模板的序号。支持 BMP 图像。
TemplateO, char*	
Template1, char*	
Template2, char*	
Template3, char*	
Template4, char*	
Template5, char*	
Template6, char*	
Template7, char*	
Template8, char*	
Template9)	
float*	图像匹配,返回值中的前 2 个元素是按顺序对应的
ImageMatching(char*	目标与模板的差异分数,最后一个元素是匹配到的
TargetImage, char*	模板的序号。支持 BMP 图像。
TemplateO, char*	
Template1)	
float*	图像匹配,如果模板数为 n,则返回值中的前 n 个元
ImageMatching(Image3	素是按顺序对应的目标与模板的差异分数,最后一

```
个元素是匹配到的模板的序号。支持 BMP 图像。
Target Image, float
                       需引入以下结构体:
**templates, int
num templates)
                       typedef struct
                           float *data;
                           int width;
                           int height;
                       } Image3;
                       声明:
                       Image3 load bmp(char *filename);
                       Image3 img, TargetImage;
                       float **templates;
                       int num_templates = 10;
                           templates
                                                    (float
                                                                 **)
                       malloc(sizeof(float *) * num templates);
                           img = load bmp(Template0);
                           templates[0] = img. data;
                           img = load bmp(Template1);
                           templates[1] = img. data;
                           img = load_bmp(Template2);
                           templates[2] = img. data;
                           img = load bmp(Template3);
                           templates[3] = img. data;
                           img = load_bmp(Template4);
                           templates[4] = img. data;
                           img = load bmp(Template5);
                           templates[5] = img.data;
                           img = load bmp(Template6);
                           templates[6] = img. data;
                           img = load bmp(Template7);
                           templates[7] = img. data;
                           img = load_bmp(Template8);
                           templates[8] = img. data;
                           img = load bmp(Template9);
                           templates[9] = img. data;
                       TargetImage = load_bmp(TargetImagefile);
                       图像特征。
void
                       kernel 文件内容样例:
ImageFeatures (char*
input, char*
                       3
kernel, char* output)
                       1
                       0 - 1 0
                       -1 5 -1
```

	其中,3表示尺寸为3*3,1表示内核的规模			
void FileWrite(char*	图像隐写之文件写入,将文本文件写入图像。支持32			
BMP, char* TXT)	位 BMP 图像。BMP 是要写入的图像文件名,TXT 是要			
	写入图像的文本文件名。			
void	图像隐写之文件写出,将文本文件从图像中取出来。			
FileWriteOut(char*	支持 32 位 BMP 图像。BMP 是要写出的图像文件名,			
BMP, char* TXT)	TXT 是写出图像后信息保存的文本文件名。			
void LBP(char*	LBP 图像特征提取。支持 PNG 图像。			
input, char* output)				
void LBP(char*	LBP 图像特征提取。radius 是取样半径(不小于1的			
input, char* output,	浮点数),pointNumbers 是取样点数(不小于 8 的			
int choice, float	整数), choice 的不同取值表示: 1.普通圆形 LBP			
radius, int	2. 旋转不变圆形 LBP 3. 等价模式圆形 LBP。支持 24			
pointNumbers)	位 BMP 图像,只有等价模式下取样点 pointNumbers			
portronumers)	可以大于8但不得大于255。			
void Watershed2(char*	图像分割之分水岭算法。inputMarqueurs 是输入图			
input, char*	像的标记图像。R=230, G=0, B=0, r=1。支持 PNG 图			
* '				
inputMarqueurs, char*	像。			
output, int r, unsigned				
char R, unsigned char				
G, unsigned char B)				
void	图像分割。rayon=5。支持 PNG 图像。			
EcrireImage1(char*				
input, char*				
output, uint32_t				
rayon)				
void	图像分割。rayon=5。支持 PNG 图像。			
EcrireImage2(char*				
input, char*				
inputMarqueurs,char*				
output, uint32_t				
rayon)				
void	图像分割。rayon=5。支持 PNG 图像。			
EcrireLPECouleur1(ch				
ar* input, char*				
inputMarqueurs, char*				
output, uint32_t				
rayon)				
void Watershedl(char*	图像分割之分水岭算法。inputMarqueurs 是输入图			
input, char*	像的标记图像。rayon=5。支持 PNG 图像。			
inputMarqueurs, char*	2004 Miles			
output, uint32_t				
rayon)				
1 0 1 / 1				

void	图像分割。rayon=1。支持 PNG 图像。
EcrireImage3(char*	
input, char*	
inputMarqueurs, char*	
output, uint16 t	
rayon)	
void	图像分割。rayon=1。支持 PNG 图像。
EcrireImageCouleursA	国家为 III I I I 文 N I I I I I I I I I I I I I
leatoires (char*	
input, char*	
= '	
inputMarqueurs, char*	
output, uint8_t	
r, uint8_t g, uint8_t	
b, uint16_t rayon)	
void Watershed(char*	图像分割之分水岭算法。inputMarqueurs 是输入图
input, char*	像的标记图像。a 一般为 255, rayon=1。支持 PNG 图
inputMarqueurs,char*	像。
output, uint8_t	
r, uint8_t g, uint8_t	
b, uint8_t a, uint16_t	
rayon)	
void FloodFill(char*	图像分割之漫水填充法。x=0,y=0,novaCor=127。
input, char*	支持 PGM 文件。
output, int x, int	) (1) (1) (1) (1) (1) (1) (1) (1) (1) (1
y, unsigned char	
novaCor)	
void	   读取点坐标并输出点之间绘制的点或线段的图像。
ConvertCoordinatesTo	IMAGE SIZE=800, PIXEL PADDING=25, drawlines=0
Graphics (char*	或 drawlines=1。 绘入文件权力
input, char* output,	输入文件格式
double IMAGE_SIZE,	如果我们将"N"表示为点数,则假设采用以下点坐
double PIXEL_PADDING,	标文件格式:
bool drawlines)	N
	1 x 坐标 y 坐标
	2 x 坐标 y 坐标
	3 x 坐标 y 坐标
	4 x 坐标 y 坐标
	N x 坐标 y 坐标
void	人类检测。参考: MINH=0.0, MAXH=50.0, MINS=0.23,
HumanDetection1(char	MAXS=0.68。支持 24 位 BMP 图像。
* input, char* output,	244 E - EM:
double MINH, double	
MAXH, double MINS,	
MAAH, GOUDTE MINS,	

double MAXS)				
void	人类检测。参考: MINH=0.0,MAXH=50.0,MINS=0.23,			
HumanDetection2(char	MAXS=0.68。支持 24 位 BMP 图像。			
* input, char*	The court of the c			
output, double MINH,				
double MAXH, double				
MINS, double MAXS)				
void	人类检测。参考: MINH=0.0, MAXH=50.0, MINS=0.23,			
HumanDetection3(char	MAXS=0.68。支持 24 位 BMP 图像。			
* input, char*				
output, double MINH,				
double MAXH, double				
MINS, double MAXS)				
int	计算图像的特征点数量。支持 24 位 BMP 图像。			
ImageFeatureNumber(c				
har* input)				
int	返回两个图像的内容相似度。			
ContentSimilarity(ch	KDTREE_BBF_MAX_NN_CHKS 是在 BBF 搜索期间要检查			
ar* input1, char*	的 关 键 点 NN 候 选 的 最 大 数 量 ,			
input2, int	NN_SQ_DIST_RATIO_THR 是 NN 和第二个 NN 之间距离			
KDTREE_BBF_MAX_NN_CH	平 方 比 的 阈 值 , 参 考 :			
KS, double	KDTREE_BBF_MAX_NN_CHKS=100 ,			
NN_SQ_DIST_RATIO_THR	NN_SQ_DIST_RATIO_THR=0.49。支持 24 位 BMP 图像。			
)				
Feature*	返回图像的特征数据。支持 24 位 BMP 图像。			
ImageFeature(char*	需引入一下结构体:			
input)	typedef struct Point2D64f			
	{			
	double x;			
	double y;			
	}Point2D64f;			
	typodof atrust footure			
	typedef struct feature			
	{   double x; /x 坐标/			
	double x; /x 坐标/ /y 坐标/ /y 坐标/			
	double a; /0xford 型			
	(方射区域参数/			
	double b; /0xford 型			
	仿射区域参数/			
	double c; /Oxford 型			
	仿射区域参数/			
	double scl; /Lowe 风格			
	特征的比例/			
	14 hrs 24 t 0 h 4/			

	double ori;	/Lowe 风格	
	, 特征的方向/	, , , , , , , , ,	
	int d;	/描述符长	
	度/		
	double descr[128];	/描述符/	
	int type;	/功能类	
	型,OXFD 或 LOWE/		
	int category;	/通用功能	
	类别/		
	struct feature* fwd_match;	/前向图像	
	的匹配特征/		
	struct feature* bck_match;	/ 从	
	backmward 图像中匹配特征/		
	struct feature* mdl_match;	/匹配模型	
	中的特征/		
	Point2D64f img_pt;	/图像中的位	
	置/		
	Point2D64f mdl_pt;	/模型中的位	
	置/		
	void* feature_data;	/用户可定	
	义数据/		
	}Feature;		
doub1e	字符匹配,支持 BMP 图像,返回值是		
CharacterRecognition	到的模板文件的序号,如返回值是 2		
(char*	序号为2(序号从零开始)的模板匹配。		
TargetImage, char*	参考: TemplateFileGroup[]={ "0.txt", "1.txt",		
TemplateFileGroup[])	"2. txt", "3. txt", "4. txt", "5. txt", "6. txt",		
1 11	"7. txt", "8. txt", "9. txt" };		
double	字符匹配,支持 BMP 图像,返回值是目标图像匹配		
CharacterRecognition	到的模板文件的序号,如返回值是 2 则说明图像与		
1 (char*	序号为 2 (序号从零开始) 的模板匹配。		
TargetImage, char*	参考: TemplateFileGroup[]={ "0. txt", "1. txt", "2. txt", "3. txt", "4. txt", "5. txt", "6. txt",		
TemplateFileGroup[])		xt, b.txt,	
	"7. txt", "8. txt", "9. txt" };		