使用说明书

目录

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

PPM、PGM 和 PBM 图像处理

input 是输入文件名,
文件名。支持 P5 格式的
BMP 图像。bpp 是 BMP 图
M 图像。
BMP 图像。bpp 是 BMP 图
M图像。
分。input 是输入文件名,
文件名。支持 P5 格式的
input 是输入文件名,
文件名。支持 P3 格式的
并返回图像数据。input
BM 图像文件名。支持 P4
像 。

:1 W: DDW/ : 1 1	四去 DDU 图像 I 、日松)始图像料
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:
	<pre>typedef struct {</pre>
	unsigned char red, green, blue;
	//像素的颜色由 RGB (红/绿/
	蓝)表示
	<pre>} PPMPixel;</pre>
	typedef struct {
	unsigned int width, height;
	// 图像的宽度和高度(以像素
	为单位)
	/ -
	PPMPixel *data;
	// 构成图像的像素
	} PPMImage;
<pre>void WritePPM(char* output, PPMImage* img)</pre>	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
	<pre>} PPMPixe1;</pre>
	typedef struct {
	unsigned int width, height;
	_
	// 图像的宽度和高度(以像素
	为单位)
	PPMPixel *data;
	// 构成图像的像素
	<pre>} PPMImage;</pre>
void InvertColor(char*	负滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 P6 格式的 PPM 图
input, onar output,	像。
void GrayFilter(char*	灰度过滤器, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P6 格式的
	PPM 图像。
void SepiaFilter(char*	乌贼墨过滤器, input 是输入文件名,

	, 目於山文供有 土柱 DC 按子的
input, char* output)	output 是输出文件名。支持 P6 格式的 PPM 图像。
void AdjustSaturation(char*	7 7 7 7 7
	调整图像饱和度, input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标饱和度,
	如 a=30。支持 P6 格式的 PPM 图像。
void Resize(char* input, char*	调整图像大小,input 是输入文件名,
output, unsigned int NewWidth,	output 是输出文件名。NewWidth 和
unsigned int NewHeight)	NewHeight 分别是输出图像的宽和高。
	支持 P6 格式的 PPM 图像。
void AdjustHue(char* input,char*	调整图像的色调,input 是输入文件名,
output, int a)	output 是输出文件名。a 是目标色调,
	如 a=125。支持 P6 格式的 PPM 图像。
void AdjustBrightness(char*	调整图像亮度, input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标亮度,
	如 a=60。支持 P6 格式的 PPM 图像。
void AdjustContrast(char*	调整图像对比度,input 是输入文件名,
input, char* output, double a)	output 是输出文件名。a 是目标对比度,
1 ,	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
impat, char. output, double a	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,
Input, char* output, double a)	
:1 D: 1 (1 at : at 1 at	如 a=3。支持 P6 格式的 PPM 图像。
void Pixelate(char* input, char*	像素化,input 是输入文件名,output
output, unsigned int a)	是输出文件名。a 是幅度值,如 a=8。
	支持 P6 格式的 PPM 图像。
void Rotate(char* input, char*	旋转图像,input 是输入文件名,output
output, short a)	是输出文件名。a 是旋转的角度,如
	a=45。支持 P6 格式的 PPM 图像。
void GammaCorrection(char*	伽马校正, input 是输入文件名, output
input, char* output, double a)	是输出文件名。a是gamma数,如a=0.5。
	支持 P6 格式的 PPM 图像。
void	生成灰度图以及 RGB 通道分离, input
GrayAndChannelSeparation(char*	是输入的 P6 格式的 PPM 图像;
input, char* Grayoutput, char*	Grayoutput 是输出的灰度图文件名,
Routput, char* Goutput, char*	Routput、Goutput 和 Boutput 分别是输
Boutput)	出的 R、G 和 B 通道的图像文件名,输
	出都是 PGM 格式文件。
void PGMBin(char* input, char*	灰度图像二值化,输入是灰度图像,输
output, int threshold)	入和输出都是 PGM 文件, threshold 是
	阈值,如 threshold=125。
void Brightening(char*	彩色图像增亮,输入和输出都是 P6 格

input, char* output, int a)	式的 PPM 图像, a 是增亮系数, 如 a=80。
void GrayBrightening(char*	灰度图像增亮,输入和输出都是 PGM 图
input, char* output, int a)	像, a 是增亮系数, 如 a=80。
void PPMFilter(char* input, char*	彩色图像滤波,输入和输出都是 P6 格
output)	式的 PPM 文件。
void PGMGrayFilter(char*	灰度图像滤波,输入和输出都是 PGM 图
input, char* output)	像。
void PPMtoBMP(char* input, char*	PPM 图像转 BMP 图像, input 是输入文
output)	件名, output 是输出文件名。支持 P6
	格式的 PPM 图像。
void PGMOtsuThreshold(string	大津阈值法, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P5 格式的 PGM 图像。
void PGMRotated(char*	channels 是输入图像的通道, theta 是
input, char* output, int width, int	旋转弧度,参考:
height, int channels, double	theta=45.0*3.1415926/180.
theta)	
void XCorner(char* input, char*	channels 是输入图像的通道。
output, int width, int height, int	
channels, double theta)	
void YCorner(char* input, char*	channels 是输入图像的通道。
output, int width, int height, int	
channels, double theta)	
void Smooth(char* input, char*	channels 是输入图像的通道,sigma_x
output, int width, int height, int	是 X 方向的模糊系数, sigma_y 是 Y 方
channels, float sigma_x, float	向的模糊系数。
sigma_y, double theta)	
void	局部大津阈值,input 是输入文件名,
PGMLocalisedOtsuThreshold(string	output 是输出文件名。支持 P5 格式的
input, char* output)	PGM 图像。
void PGMSauvolaThreshold(string	索沃拉阈值,支持 P5 格式的 PGM 图像。
input, char* output, double	a 、 b 和 c 的 参 考 值 如 :
a, double b, double c)	a=0.01, b=15, c=225。 阈值法,input 是输入文件名,output
void PGMThreshold(string input, char* output, int thresh)	國祖法,Input 是制入文件名,output 是输出文件名。支持 P5 格式的 PGM 图
input, chair output, int thresh)	像。thresh 是阈值,如:thresh=5。
float Repair1(char* input, char*	图像修复, var 是噪声方差, threshold
output, float var, float	是阈值, nbLevels 是要处理的级别数,
threshold, int nbLevels, float a)	a=10。返回 ISNR。
float Repair2(char* input, char*	图像修复, var 是噪声方差, threshold
output, float var, float	是阈值, nbLevels 是要处理的级别数,
threshold, int nbLevels, float a)	a=10。返回 ISNR。
void LowPassFilterRepair1(char*	低通滤波图像修复, a=10, b=6,
input, char* output, int	nbLevels=3,size_filter 是低通过滤

size_filter, float var, int	器的大小, var 是噪声方差,
nb_iterations, int nbLevels, float	nb_iterations 是 Landweber 的迭代
a, int b)	数。
void LowPassFilterRepair2(char*	低通滤波图像修复, a=10, b=6,
input, char* output, int	nbLevels=3,size_filter 是低通过滤
size_filter, float var, int	器的大小, var 是噪声方差,
nb_iterations, int nbLevels, float	nb_iterations 是 Landweber 的迭代
a, int b)	数。
float LowPassFilterRepair3(char*	低通滤波图像修复, a=10, b=6,
input, char* output, int	nbLevels=3, pas=1, size_filter 是低
size_filter, float var, int	通过滤器的大小, var 是噪声方差,
nb_iterations, int nbLevels, int	nb_iterations 是 Landweber 的迭代
pas, float a, int b)	数。返回 ISNR。
void Repair1(char* input, char*	图像修复, a=0.0, M 是分解的层次数,
output, int M, float a)	如 M=3。
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)	

RAW 图像处理

unsig	ned	char** R	AWRead(char*	RAW 图像读取,返回像素数据。
input	int hei,	ght, int widt	h)	
void	RAWW	rite(unsigne	d char**	接收像素数据保存为 RAW 图像。
input	, char*	output, int	height, int	
width)			
void	MBV	Q(char*	input,char*	MBVQ 效果, input 是输入文件名,
outpu	t,int wi	dth,int heig	ht)	output 是输出文件名。width 和
				height 是输出图像的宽和高。
void	RAWtoPP	M_red(char*	input,char*	RAW 转为 PPM 后提取红色通道,

output, int width, int	参考: width=4096, height=3072,
,	
height, DebayerAlgorithm algo)	
	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,
input, char* output, int width, int	参考: width=4096, height=3072,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或
neight, bedayernigorithm argor	LINEAR.
	支持 RAW12 格式。
	1 - 1 - 1
	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,
input, char* output, int width, int	参考: width=4096, height=3072,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或
	LINEAR。
	支持 RAW12 格式。
	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
void RAWtoPPM_blue(char* input,char*	RAW 转为 PPM 后提取蓝色通道,
_	参考: width=4096, height=3072,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或
	LINEAR .
	支持 RAW12 格式。
	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
void RAWtoPPM(char* input, char*	RAW 转为 PPM,参考: width=4096,
output, int width, int	height=3072 ,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或
	LINEAR。
	支持 RAW12 格式。
	需引入以下枚举:
	104 417 421 1 124 1 4

	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
void RawPowerTransformation(char*	幂次变换, input 是输入的 RAW 图
input, char* output, int width, int	像文件名,output 是输出的 RAW
height, int c, float v)	图像文件名,width 是输入图像的
	宽,height 是输入图像的高。默
	认 c=1, v=0.6。支持 RAW 图像。
void RAWAvgFilter(char* input,char*	平均滤波器, input 是输入文件
	名,output 是输出文件名。ROWS
output, int ROWS, int COLS, int M, float	
mask[3][3])	是图像的行大小,COLS 是图像的
	列大小, M 是滤波相关参数, 如
	M=1; mask 是滤波器模板。支持
	RAW图像。
	参考模板:
	float mask[3][3] =
	{{0.1111, 0.1111, 0.1111}},
	{0.1111, 0.1111, 0.1111},
	$\{0.1111, 0.1111, 0.1111\}\};$
void RawImageInversion(char*	图像反相, input 是输入的 RAW 图
input, char* output, int width, int	像文件名,output 是输出的 RAW
height)	图像文件名,width 是输入图像的
ne 1811e)	宽,height 是输入图像的高。支
	持 RAW 图像。
void RawHistogramEqualization(char*	直方图均衡化, input 是输入的
	RAW 图像文件名, output 是输出
input, char* output, int width, int	
height)	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。
void RAWHistogramEqualization(char*	RAW 直方图均衡化, width 和
input, char* output, int width, int	height 是输入图像的宽和高。
height)	
void RAWMedianFilter(char*	中值滤波, input 是输入文件名,
input, char* output, int ROWS, int	output 是输出文件名。ROWS 是图
COLS, int M, int sequence[9])	像的行,COLS 是图像的列,M 是
	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
output, unsigned long Width, unsigned	输入文件名,output 是输出文件
long Height)	名。Width 和 Height 是输入文件
	的宽和高。
void RawToBmp(char* input, char*	RAW 图像转为 BMP 图像, input 是
output, int imageWidth, int imageHigth)	输入文件名, output 是输出文件
output, into imagenitatin, into imagenitatin,	名。支持宽和高相等的图像。
void RGBtoCMY(string input, string	RGB 转 CMY, output 1 是输出的 CMY
output1, string output2, int height, int	模型图像名,output2 是输出的
width, int NumberChannels, int a)	CMY 模型图像的单个通道的图像
	名; height 和 width 是输入图像
	input的高和宽;NumberChannels
	是图像的通道数,如通道数为3;
	a=0表示生成 cyan 模型图像,a=1
	表示生成 magenta 模型图像, a=2
	表示生成 yellow 模型图像。支持
	RAW 文件。
void RGBtoHSI(char* input, char*	RGB 模型转为 HIS 模型, input 是
output)	输入文件名, output 是输出文件
output)	
.1 0.1.101/1	名。支持 24 位 BMP 图像。
void SobelOperation1(char*	Sobel 算子。
input, char* output, int width, int	
height)	
void SobelOperation2(char*	Sobel 算子。
input, char* output, int width, int	
height)	
void CyanGray(char* input, char*	青色灰度图像。
output, int width, int height)	
void MagentaGray(char* input, char*	品红灰度图像。
output, int width, int height)	
void YellowGray(char* input, char*	黄色灰度图像。
output, int width, int height)	
void GrayLightness(string	 彩色转灰度。
input, string output, int height, int	/IV LITY //\/X
width, int NumberChannels)	彩色柱 左座
void GrayAverage(string input, string	彩色转灰度。
output, int height, int width, int	
NumberChannels)	5/ F 1/ 1- P
void GrayLuminosity(string	彩色转灰度。
input, string output, int height, int	
width, int NumberChannels)	
void Transfer(char* input, char*	传递函数。
output, int width, int height)	
void Homography(char* input1, char*	单应。
3 1 1 ,	I .

input2, char* input3, char* output, int	
width, int height, int newwidth, int	
newheight)	
void MovieEffect(char* input, char* 电影效	果。
output, int width, int height)	
void Dither(string input, string 抖动,	参考: method=1,
output, int height, int width, int bayerMa	atrixNumber=2 ,
NumberChannels, int method, int numberC	OfTones=2。
bayerMatrixNumber, int numberOfTones)	
void AssimilateChannels(string 抖动,	参考: method=1 ,
	atrixNumber=2 ,
	OfTones=2。
method, int bayerMatrixNumber, int	
numberOfTones)	
,	
input, char* output, int width, int	
height)	
input, char* output, int width, int	= //C // 0 // 4 /E / D /
height)	
	理,抖动矩阵法,默认 N=2。
input, char* output, int width, int	
height, int N)	
	與, 规范化对数。
input, char* output, int width, int	
height)	
	换,规范化对数。
input, char* output, int width, int	
height)	
void TernaryGrayLevel1(char* 三值灰)	· · · · · · · · · · · · · · · · · · ·
input, char* output, int width, int	
height)	
void TernaryGrayLevel2(char* 三值灰)	· .
input, char* output, int width, int	
height)	
void BestEdgeMap1(char* input,char* 最佳边界	站图 。
output, int width, int height)	
void BestEdgeMap2(char* input,char* 最佳边见	·····································
output, int width, int height)	
void Skeletonize(char* input, char* 骨架化。	
output, int width, int height)	
void GrayLuminosity(string 转换为	灰度。
input, string output, int height, int	
width, int NumberChannels)	

	/b/ /\ [급] //c \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
void	微分图像以获得边缘。
DifferentiateImageSobelFilter(string	
input, string output, int height, int	
width, int NumberChannels)	
void	求反图像以获得边缘映射。
DifferentiateImageSobelFilterAndRGBto	
CMY (string input, string output, int	
height, int width, int NumberChannels)	
void EdgeDetectionSobelFilter(string	使用 Sobel 过滤器获取图像的边
input, string output, int height, int	缘贴图,参考: threshold=50。
width, int NumberChannels, int	3,7,11,11,11,11,11,11,11,11,11,11,11,11,1
threshold)	
void RemoveSpeckles(string	参考: threshold=-100,
input, string output, int height, int	hackground=255
width, int NumberChannels, int	numberOfIterations=0,
threshold, int background, int	morphingOperation=THINNING.
numberOfIterations, type_morphing	本函数需添加以下结构体:
morphingOperation)	typedef enum type_morphing {
	SHRINKING = 0x1,
	THINNING = 0x2,
	SKELETONIZING = 0x3
	} type_morphing;
void	双微分图像获取边缘。
DoubleDifferentiatingGetEdges(string	
input, string output, int height, int	
width, int NumberChannels)	
width, int NumberChannels) void GetTernaryMap(string	创建三元边贴图。参考:
	创建三元边贴图。参考: threshold=50 ,
void GetTernaryMap(string	
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int	threshold=50 ,
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string	threshold=50 ,
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool	threshold=50 ,
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile)	threshold=50 , writeHistogramToFile=false。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char*	threshold=50 ,
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int	threshold=50 , writeHistogramToFile=false。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height)	threshold=50 , writeHistogramToFile=false。 可分离扩散。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char*	threshold=50 , writeHistogramToFile=false。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int	threshold=50 , writeHistogramToFile=false。 可分离扩散。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height)	threshold=50 , writeHistogramToFile=false。 可分离扩散。 去除噪声。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char*	threshold=50 , writeHistogramToFile=false。 可分离扩散。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height)	threshold=50 , writeHistogramToFile=false。 可分离扩散。 去除噪声。 - 売度调整。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char*	threshold=50 , writeHistogramToFile=false。 可分离扩散。 去除噪声。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char*input, char* output, int width, int height) void Denoising(char* input1, char*input2, char* output, int width, int height) void Luminosity(char* input, char*output, int width, int height) void Average(char* input, char*output, int width, int height)	threshold=50 , writeHistogramToFile=false。 可分离扩散。 去除噪声。 亮度调整。 平均化。
void GetTernaryMap(string input, string output, int height, int width, int NumberChannels, int threshold, string output_Histogram, bool writeHistogramToFile) void SeparableDiffusion(char* input, char* output, int width, int height) void Denoising(char* input1, char* input2, char* output, int width, int height) void Luminosity(char* input, char* output, int width, int height) void Average(char* input, char*	threshold=50 , writeHistogramToFile=false。 可分离扩散。 去除噪声。 - 売度调整。

void Shrink(char* input, char*	收缩。
output, int width, int height)	
<pre>void BilinearTransformation(char*</pre>	双线性变换。
input, char* output, int width, int	
height, int newwidth, int newheight)	
void BilinearInterpolation(string	双线性插值,参考:
input, string output, int height, int	targetHeight=200 ,
width, int NumberChannels, int	targetWidth=200。
targetHeight, int targetWidth)	
void DitherMatrixMethod(char*	四级抖动,默认 N=2。
input, char* output, int width, int	
height, int N)	
int ObjectsInImages(string	图像中的对象,参考:
input, string output, int height, int	threshold=128 ,
width, int NumberChannels, int	starSize=1.5833, a=1 表示确保
threshold, float starSize, int a)	前背景为黑色。支持 RAW 文件。
void Dewarped1(char* input, char*	脱蜡。a是在输出图像中检查半径
output, int width, int height, int	是否<=a,然后再进行扭曲,参考:
Offset, double a, double b)	Offset=256, a=256.5, b=0.5。
void Dewarped2(char* input, char*	脱蜡。a 是在输出图像中检查半径
output, int width, int height, int	是否<=a,然后再进行扭曲,参考:
Offset, double a, double b, double	Offset=256, a=256.5, b=0.5.
coeffx[12], double coeffy[12])	脱蜡规范:
	double coeffx[12] =
	{ 1.00056776e+00, -
	5. 68880703e-04,
	1. 13998357e-03,
	1.00056888e+00,-
	5. 65549579e-04, -1. 13554790e-
	03,
	9. 99434446e-
	9. 99434440e 01 , 5. 66658513e-04 ,
	1.13110351e-03,
	9. 99433341e-
	9. 99433341e 01 , 5. 67767429e-04 ,
	1. 13553921e-03 };
	1.100000216 00),
	double coeffy[12] = {-
	5. 67763072e-04,
	1. 00056888e+00, 1. 13998357e-
	03,
	5. 68880703e-
	04, 9.99434450e-01, -
	01, 0.001011000 01,

	1 101-0
	1. 13554790e-03,
	5. 65553919e-
	04, 9.99433341e-01, -
	1.13110351e-03,
	-5.66658513e-
	04, 1.00056777e+00,
	1. 13553921e-03};
void TextureSegmentation1(char*	纹理分割,默认 K=6, N=100。
input, char* output, int width, int	20-11 H
height, int K, int N)	
void TextureSegmentation2(char*	纹理分割,默认 K=6, N=100。
input, char* output, int width, int	
height, int K, int N)	
void TextureClassification(vector	纹理分类,a 是要分类的图像的数
<pre><string> filename, char* output, int</string></pre>	量,如 filename 里有 3 个图像名
width, int height, int K, int N, int a)	称,则 a=3; output 是分类结果
	文件,格式为 txt 的文本文件;
	默认 K=4, N=1000。
void ErrorDiffusion1(char*	误差扩散。
input, char* output, int width, int	
height)	
void ErrorDiffusion2(char*	误差扩散。
input, char* output, int width, int	70.13V
height)	
void ErrorDiffusion3(char*	
input, char* output, int width, int	(人在J) 版。
height)	
	提美比斯
void ErrorDiffusion(string	
input, string output, int height, int	kernelSize=3 ,
width, int NumberChannels, int	numberOfTones=2 ,
method, int kernelSize, int	useFilter=false。支持 RAW 文
numberOfTones, bool useFilter)	件。
void Thin(char* input, char*	图像细化。
output, int width, int height)	
void	形态滤波。
BinaryMorphologicalFilteringComplete(参考: threshold=100 ,
string input, string output, int	numberOfIterations=0 ,
height, int width, int	morphingOperation=(type_morp
NumberChannels, int threshold, int	hing)3。支持 RAW 文件。
numberOfIterations, type_morphing	
morphingOperation)	
void OilPainting(char* input, char*	油画效果,默认 N=2。
output, int width, int height, int N)	
void OilPainting1(char* input, char*	油画效果,默认 N=2。
. or	TH F-1/2/1/1 MYL 9/L 11 40

void OilPaintingEffect1(string 油	
void	油画效果,参考: colorBits=3,
input, string output, int height, int b	oitsOK=true, kernelSize=9,
width, int NumberChannels, int ke	kernelSizeOK=true。支持 RAW 文
colorBits, bool bitsOK, int 性	件 。
kernelSize, bool kernelSizeOK)	
void 0ilPaintingEffect2(string 油	油画效果,参考: colorBits=3,
	oitsOK=true, kernelSize=9,
	kernelSizeOK=true。支持RAW文
	件。
kernelSize, bool kernelSizeOK)	
	油画效果,参考: colorBits=3,
input, string output, int height, int b	oitsOK=true, kernelSize=9,
	kernelSizeOK=true。支持 RAW 文
	件 。
kernelSize, bool kernelSizeOK)	
	3*3 平均值滤波。
inputfile, char* outputfile, int	
width, int height)	
void GeometricMeanFiltering(char* 3	3*3 几何均值滤波。
inputfile, char* outputfile, int	
width, int height)	
void MedianFiltering(char* 中	中值滤波。
inputfile, char* outputfile, int	
width, int height)	
void FFT(char* input, char* output, int FI	FFT 函数。
width, int height)	
void LowPassOrHighPassFiltering(char* 但	低通或高通滤波。LOW_PASS=1为
input, char* output, int width, int 仅	低通滤波,否则为高通滤波,
height, int LOW_PASS, int DEGREE)	DEGREE 为滤波程度,如
DI	DEGREE=0。
void IFFT(char* input, char* II	IFFT 函数。LOW_PASS=1 为低通滤
output, int width, int height, int 波	波,否则为高通滤波,DEGREE 为
LOW_PASS, int DEGREE) 滤	滤波程度,如 DEGREE=0。
void BMPtoRAW(char* inputfile,char* BMPtoRAW)	BMP 图像转 RAW 图像。支持 24 为
outputfile) Bi	BMP 图像。
void BMPtoRAW1(char* input, char* B	BMP 图像转 RAW 图像。支持 24 为
output) Bi	BMP 图像。

BMP 图像处理

unsigned char**	读取 8 位 BMP 图像的像素。
BMPRead8(char* input)	
void GenerateImage8(char*	生成 8 位 BMP 图像, output 是生成的图像
output, unsigned char**	文件名, color 是像素数据。

color)	
BMPMat** BMPRead(char*	读取 24 位和 32 位 BMP 图像的像素。
input)	需要引入以下结构体:
-	typedef struct {
	unsigned char B; //24位和 32位BMP
	图像的蓝色通道分量
	unsigned char G; //24 位和 32 位 BMP
	图像的绿色通道分量
	unsigned char R; //24位和32位BMP
	图像的红色通道分量
	unsigned char A; //仅限 32位 BMP 图
	像的 Alpha 通道
	}BMPMat;
unsigned int BMPHeight(char*	读取 BMP 图像的高度。
input)	
unsigned int BMPWidth(char*	读取 BMP 图像的宽度。
input)	
void GenerateImage(char*	生成 24 位和 32 位 BMP 图像。type 等于图
output,BMPMat**	像的位数,如 type=24。
color, unsigned short type)	参考用例:
	BMPMat** color =
	(BMPMat**)malloc(sizeof(BMPMat*)*1280
);
	for (unsigned int i = 0; i < 1280;
	i++)
	{
	color[i] =
	(BMPMat*)malloc(sizeof(BMPMat)*2450);
	}
	for (unsigned int i = 0; i < 1280;
	i++) {
	for (unsigned int $j = 0$; $j < 0$
	2450; j++) {
	color[i][j].B =0;
	color[i][j].G =0;
	color[i][j].R =255; }
	}
void	直方图均衡,支持8位和16位BMP。input
HistogramEqualization5(char*	是输入文件名,output 是输出文件名。
input, char* output)	左前/八人川 口, Output 定側山人 石。
void Resize(char*	图片缩放,支持 8 位和 16 位 BMP。input 是
input, char* output, int	输入文件名, output 是输出文件名。Height
Height, int Width)	和 Width 是输出图像的高和宽。
11010110, 1110 "114011/	□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

1 11 W D 1 (1)	子圆海站亚北京库 十柱 0 份和 1 C 份 DUD
double MeanBrightness(char*	求图像的平均亮度,支持8位和16位BMP。
input)	input 是输入文件名。
int IsBitMap(FILE *fp)	判断是否是位图。
int getWidth(FILE *fp)	获得图片的宽度。
int getHeight(FILE *fp)	获得图片的高度。
unsigned short getBit(FILE	获得每个像素的位数。
*fp)	
unsigned int getOffSet(FILE	获得数据的起始位置。
*fp)	
void BMPtoTIFF(char*	BMP 图像转 TIFF 图像。
input, char* output)	
void BMPtoYUV(char*	BMP 图像转为 YUV 图像, input 是输入文件
input, char* output, char	名,output 是输出文件名。yuvmode 是 YUV
yuvmode)	文件的 3 个模式选项, yuvmode 的值可为
	'0'、'2'、'4',分别为 420, 422, 444
void BMPLoadedIntoYUV(char*	YUV 加载 BMP, inputBMP 是输入的 BMP 图
inputBMP, char*	像,inputYUV 是输入的YUV 图像,inputYUV
inputYUV, char* output, int	起到容器的作用,YUVwidth和YUVheight是
YUVwidth, int YUVheight, int	输入的 YUV 图像的宽和高,参考: depth=12,
depth, bool mt)	mt=true。
void BMPtoYUV420I(char*	BMP 图像转为 YUV420 图像, input 是输入文
input, char* output)	件名, output 是输出文件名。
void BMPtoYUV420II(char*	BMP 图像转为 YUV420 图像, input 是输入文
input, char* output)	件名, output 是输出文件名。
void DCMtoBMP(string	DCM 图像转 BMP 图像。input 是输入文件名,
input, char* output)	output 是输出文件名。
void Ins1977 (char*	
input, char* output, int	是输出文件名。参考: ratio=100。
ratio)	Children of the second of the
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)	м. п. э J. 14010 1000
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,
a, 404510 0/	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。
inputt, chair inputz, chair	JHL10

output)	年八叶光 1 1 100 0 1 0 5
void Short(char* input, char*	矮化特效。a=1, b=128, c=2, d=0.5,
output, int a, int b, int	depth=24。支持 24 位 BMP 图像。
c, double d, int depth)	
void Rise(char* input, char*	增高特效。a=1, b=128, c=0.5, d=2,
output, int a, int b, double	depth=24。支持 24 位 BMP 图像。
c, int d, int depth)	
void Short1(char*	矮小化特效。a=1, b=128, c=0.5, d=0.5,
input, char* output, int a, int	depth=24。支持 24 位 BMP 图像。
b, double c, double d, int	
depth)	
void Handstand(char*	倒立特效。a=1,b=128,c=0.5,depth=24。
input, char* output, int a, int	支持 24 位 BMP 图像。
b, double c, int depth)	7,11 II I
void Fat (char* input, char*	肥胖特效。a=1, b=128, c=0.5, depth=24。
output, int a, int b, double	之持 24 位 BMP 图像。
- · · · · · · · · · · · · · · · · · · ·	文元 4年 DIMI 国家。
c, int depth)	宣明性效1 1-1000 1-0 5
void HighFoot(char*	高脚特效。a=1, b=128, c=2, d=0.5,
input, char* output, int a, int	depth=24。支持 24 位 BMP 图像。
b, int c, double d, int depth)	
void CurvedCurve(char*	弧度弯曲特效。a=1,b=128,c=4,d=2,e=0.5,
input, char* output, int a, int	depth=24。支持 24 位 BMP 图像。
b, int c, int d, double e, int	
depth)	
void Thin(char* input, char*	细化特效。a=1, b=128, c=0.5, d=0.5,
output, int a, int b, double	depth=24。支持 24 位 BMP 图像。
c, double d, int depth)	
void Winding(char*	弯曲特效。lim=20, a=1, b=128, c=4, d=5,
input, char* output, int	e=0.5, depth=24。支持 24 位 BMP 图像。
lim, int a, int b, int c, int	-
d, double e, int depth)	
void ImageFusion3(char*	多聚焦图像的融合,支持 8 位 BMP 图像。
input1, char* input2, char*	block height=8, block width=8,
output, int block height, int	threshold=1.75.
block_width, double	chi conora 1.100
threshold)	
	夕取住风俛的融合 古柱 Q 份 DMD 图佈
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)	国体引入 台水 0111 201 00 201
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

1 [] . , 1 [] . , 1	n# " , " " "
dx[], int dy[], int a, double	膜.png", output="output.jpg"。
b1, int DX1, int DY1, double	int $dx[] = \{0, 0, -1, 1\};$
EPS)	int $dy[] = \{-1, 1, 0, 0\};$
void Dark(char* input, char*	暗调滤镜,参考: ratio=100。
output, int ratio)	
void WaveFilter(char*	波浪形变特效滤镜, degree 是滤镜程度(波
input, char* output, int	浪扭曲度)。 a=0 时生成 BMP 图像, a=1 时生
degree, int a)	成 JPG 图像,a=2 时生成 PNG 图像,a=3 时
	生成 TGA 图像,参考: degree=10。
void PinchFilter(char*	挤压形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int a)	a=1 时生成 JPG 图像,a=2 时生成 PNG 图像,
	a=3 时生成 TGA 图像。
void PinchFilter(char*	挤压形变特效滤镜,a=0时生成BMP图像,
input, char* output, int	a=1 时生成 JPG 图像,a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
	坐标, cenY 是形变中心点 Y 坐标。
void SpherizeFilter(char*	球面形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int a)	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
2 ,	a=3 时生成 TGA 图像。
void SpherizeFilter(char*	球面形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
	坐标, cenY 是形变中心点 Y 坐标。
void SwirlFilter(char*	旋转形变特效滤镜,a=0时生成BMP图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
ratio, int a)	a=3 时生成 TGA 图像,ratio=3。
void SwirlFilter(char*	旋转形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int ratio, int	a=3 时生成 TGA 图像,ratio=3,cenX 是形
a)	变中心点 X 坐标, cenY 是形变中心点 Y 坐
	标。
void ClosedOperation(char*	闭运算, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 4 位 BMP 图像。
void AdjustPixel(char*	调整像素值, input 是输入文件名, output
input, char* output, int a)	是输出文件名。a 是用于设置图像像素的相
input, chai. Surput, int a/	关参数,如 a=3。支持 24 位 BMP 图像。
void	部分颜色保留滤镜,参考: ratio=60。
PartialColorRetention(char*	HE 77 DV CT NEW MINDER DE 7 . 1 I I I I I I O O O O
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 OTSU(char* input, char*	大津算法, input 是输入文件名, output 是
output, int BeforeThreshold)	输出文件名。BeforeThreshold 是初始國
output, int before in eshord)	值,如 BeforeThreshold=10。支持 8 位 BMP
	图像。
:1 I .D.: 14. (1	
void LowerBrightness(char*	调低亮度, input 是输入文件名, output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100, b=0。
void HightBrightness(char*	调高亮度,input 是输入文件名,output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100, b=0。
void	迭代阈值选择, input 是输入文件名,
IterativeThresholdSelection(output 是输出文件名。支持 8 位 BMP 图像。
char* input, char* output)	
void DitheringMethod(char*	抖动法,input 是输入文件名,output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void LogTransformation(char*	对数变换, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。constant
constant)	是相关参数,如 constant=15。
void	对数变换, input 是输入文件名, output 是
LogarithmicTransformation(ch	输出文件名。支持 BMP 图像。
ar* input, char* output)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization(char*	output 是输出文件名。支持 BMP 图像。
input, char* output)	
void Binarization(char*	二值化, input 是输入文件名, output 是输
input, char* output, int	出文件名。支持 24 位 BMP 图像。threshold
threshold)	是阈值,如: threshold=128。
void Expansion(char*	二 值 图 像 膨 胀 , 参 考 :
input, char* output, unsigned	mask[9]={0,255,0,255,255,255,0,255,0}
char mask[9], int c)	, c=128。
void Corrosion(char*	二 值 图 像 腐 蚀 , 参 考 :
input, char* output, unsigned	mask[9]={0,255,0,255,255,255,0,255,0}
char mask[9], int c)	, c=128°
void OpenOperation(char*	二值图像开运算,参考:
input, char* output, unsigned	mask[9]={0,255,0,255,255,255,0,255,0}
char mask[9], int c)	, c=128°
void ClosedOperation(char*	二值图像闭运算,参考:
input, char* output, unsigned	mask[9]={0,255,0,255,255,255,0,255,0}
char mask[9], int c)	, c=128。
void	二值图像开运算提取轮廓,参考:

	. [0] (0 0 0 0 0 0
OpenOperationToExtractContou	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
r(char* input, char*	, c=128。
output, unsigned char	
mask[9], int c)	
void	二值图像膨胀运算提取轮廓,参考:
ExpansionOperationToContourE	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
xtraction(char* input, char*	, c=128°
output, unsigned char	
mask[9], int c)	
void	二值图像腐蚀运算提取轮廓,参考:
CorrosionCalculationToContou	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
rExtraction(char*	, c=128°
input, char* output, unsigned	
char mask[9], int c)	
void Glaw(char* input, char*	发光滤镜,参考: ratio=100。
output, int ratio)	7-7-100 PU 7 3 1 100 10 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]。
void BoxBlurAdvanced(string	高级方框模糊,参考: radius=5。支持 PNG
input, string output, int	文件。
radius)	
void	高斯滤波,支持 PNG 文件。
GaussianBlurFilter(char*	
input, char* output)	
void GaussianFiltering(char*	高斯滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
1 /	

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,参考: threshold1、threshold2和
input, char* output, int	threshold3 都等于 128, color1=255,
width, int height, unsigned	color2=0。支持 24 位 BMP 图像。
char threshold1, unsigned	
char threshold2, unsigned	
char threshold3, unsigned	
char color1, unsigned char	
color2)	
void BlackWhite(char*	黑白化, input 是输入的原图像, output 是
input, char* output)	输出的黑白图像。支持 24 位 BMP 图像。
void Underexposure(char*	图像欠曝光, input 是输入的原图像,
input, char* output)	output 是输出的欠曝光图像。支持 24 位
,	BMP图像。
void Overexposure (char*	图像过曝光, input 是输入的原图像,
input, char* output)	output 是输出的过曝光图像。支持 24 位
	BMP 图像。
void Nostalgia(char*	怀旧滤镜, input 和 Mask 都是输入的文件
input, char* Mask, char*	名,Mask 是褶皱图像路径,ratio=100。
output, int ratio)	州口水松 ・ 日松 シ マ ル セ
void GammaTransform(char*	伽马变换, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void GrayScale(char*	灰度化, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void	灰度图二值化,bit 用于设定位数,如
GrayImageBinarization(char*	bit=8; threshold 是阈值,如
input, char* output, int bit, int threshold)	threshold=200。支持 8 位 BMP 图像。
void GreyPesudoColor(char*	灰度图伪彩色化, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
void HoughTransform(char*	霍夫变换, input 是输入的 RAW 文件, output
input, char* output, unsigned	是输出的 RAS 文件, threshold=100, a=800,
input, chair output, unsigned	た側山町 IAO 文下, till esilotu-100, a-000,

char threshold, int rhol, int	b=255, rho1=-295, theta1=129, rho2=-171,
thetal, int rho2, int	theta2=51, rho3=312, theta3=14.
	theta2-31, 1h03-312, theta3-14.
theta2, int a unsigned shorth	
theta3, int a, unsigned char b)	上/归.从别
static void	边缘检测,参考: a=0.33, b=0.33, c=0.33。
EdgeDetectionWithoutNonmaxim	支持 24 位 BMP 图像。
um(const LPCTSTR input, const	
LPCTSTR output, double	
a, double b, double c)	
static void	参考: a=0.33, b=0.33, c=0.33。支持24位
NonmaximumWithoutDoubleThres	BMP 图像。
holding(const LPCTSTR input,	
const LPCTSTR output, double	
a, double b, double c)	
static void	边缘检测,参考: orank=20, oranb=80。支
CannyEdgeDetection(const	持 24 位 BMP 图像。
LPCTSTR input, const LPCTSTR	
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,
HoughTransform(const LPCTSTR	orank=20, oranb=80。支持 24 位 BMP 图像。
input, const LPCTSTR	
output, double a, double	
b, double c, int orank, int	
oranb)	
void BoxBlurBasic(string	基础方框模糊,支持 PNG 文件。
input, string output)	
void	计算累加直方图并映射, input 是输入文件
CalculateCumulativeHistogram	名,outfile 是输出文件名。支持 24 位 BMP
Map(char* input, char*	图像。
outfile)	
void Translation(string	图像平移, input 是输入的文件, dx 和 dy
input, char* output, int	是横向及纵向的移动距离(像素),负值是
dx, int dy)	向左 / 向下移动; output 是平移操作后的
	结果文件名。支持 BMP 图像。
void Mirrored(string	镜像变换, input 是输入的文件, output 是
input, char* output, char	镜像操作后的结果文件名, axis 是镜像变
axis)	换的方向(以 X 或 Y 表示)。支持 BMP 图像。
void Sheared(string	错切变换, input 是输入的文件, output 是
input, char* output, char	错切操作后的结果文件名, axis 和 Coef 分
axis, double Coef)	别是错切变换的方向(以 X 或 Y 表示)和错
	切系数,负值是向左 / 向下偏移。支持 BMP
	图像。
L	1 * ***

void Scaled(string	縮放操作, input 是输入的文件, output 是
input, char* output, double	缩放操作后的结果文件名, cx 和 cy 分别是
cx, double cy)	横向及纵向的缩放系数,系数大于1表示拉
	伸,小于1表示压缩。支持BMP图像。
void Rotated1(string	图像旋转, input 是输入的文件, output 是
input, char* output, double	图像旋转后的结果文件名, angle 是旋转角
angle)	度, 弧度制。支持 BMP 图像。
void SaltNoise(char*	添加椒盐噪声,a和b是噪声相关参数,如
input, char* output, int a, int	a=3, b=3; c 和 d 是颜色相关参数, 如 c=0,
b, int c, int d)	d=255。支持 8 位 BMP 图像。
void CrossProcess(char*	交叉冲印滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void MeanFiltering(char*	均值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void MeanFlteringl(char*	均值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
void KapoorAlgorithm(char*	卡普尔算法, input 是输入文件名, output
input, char* output, int	是输出文件名。BeforeThreshold 是初始阈
BeforeThreshold)	值,如 BeforeThreshold=150。支持 8 位 BMP
	图像。
void OpenOperation(char*	开运算, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 4 位 BMP 图像。
void Diffusion(char*	扩散滤镜,参考: ratio=90。
input, char* output, int	
ratio)	
void LapulasFiltering(char*	拉普拉斯滤波, readPath 是原图像,
readPath, char*	writePath 是处理后的图像文件名。支持 8
writePath, float	位 BMP 图像。
CoefArray[9], float coef)	各参数参考值:
	定义*3 的模板 (拉普拉斯):
	float
	CoefArray[9]={1.0f, 2.0f, 1.0f, 2.0f, 4.0
	f, 2. 0f, 1. 0f, 2. 0f, 1. 0f};
	定义模板前乘的系数 (拉普拉斯):
	float coef=(float) (1.0/16.0);
void ImageFiltering(char*	图像滤波, input 是输入文件名, output 是
input, char* output, float	输出文件名。kernel 是模糊内核。支持 24
kerne1[3][3])	位 BMP 图像。
void ComicStrip(char*	连环画滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void	亮度对比度调节,参考: brightness=-30,
BrightnessAdjustment1(char*	contrast=100.

input, char* output, int	
brightness, int contrast)	
void	亮度对比度调节,参考: brightness=-30,
BrightnessAdjustment2(char*	contrast=100.
input, char* output, int	Contrast 1000
brightness, int contrast)	
void	零填充与对称扩展,支持 8 位和 24 位 BMP
ZeroFillingSymmetricExtensio	图像。
n (char* input, char* output)	
void PopArtStyle(char*	流行艺术风滤镜,参考: ratio=100。
input, char* output, int	がには 2017/14版 62、 多・9・1 att10 100。
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)	口,mask 定物/山大水区场,14110-30。
void LinearFiltering(char*	 线性滤波,input 是输入文件名,output 是
input, char* output, short	输出文件名。支持8位BMP图像。
average[3][3])	参考模板:
average[0][0]/	short average[3][3] = {{1, 2, 1},
	{2, 4, 2},
	$\{1, 2, 1, 2\},\$
void MedianFiltering(char*	中值滤波, input 是输入文件名, output 是
input, char* output, short	输出文件名。支持8位BMP图像。
average[3][3])	参考模板:
average[0][0]/	short average[3][3] = $\{\{1, 2, 1\},$
	{2, 4, 2},
	{1, 2, 1}};
void	锐化滤波, input 是输入文件名, output 是
SharpeningFiltering(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output, short	参考模板:
average[3][3], short	short average[3][3] = $\{\{1, 2, 1\},$
sharpen[3][3])	$\{2, 4, 2\},$
	$\{1, 2, 1\}\};$
	short sharpen[3][3] = $\{\{-1, -1, -1\},$
	$\{-1, 8, -1\},$
	$\{-1, -1, -1\}\};$
void	梯度锐化, input 是输入文件名, output 是
GradientSharpening(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output, short	参考模板:
average[3][3], short	short average[3][3] = $\{\{1, 2, 1\},$
soble1[3][3], short	$\{2, 4, 2\},\$
soble2[3][3])	$\{1, 2, 1\}\};$
	short soble1[3][3] = $\{\{-1, -2, -1\},$
	$\{0, 0, 0\},\$
GradientSharpening(char* input, char* output, short average[3][3], short soble1[3][3], short	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}, {1, 2, 1}}; short soble1[3][3] = {{-1, -2, -1},

void	算术平均滤波器, input 是输入文件名,
ArithmeticMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	几何平均滤波器, input 是输入文件名,
GeometricMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	调和平均滤波器, input 是输入文件名,
HarmonicMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	反调和平均滤波器, input 是输入文件名,
ContraHarmonicMeanFilter(cha	output 是输出文件名。支持 8 位 BMP 图像。
r* input, char* output)	
void Filter(char*	滤波, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 8 位 BMP 图像。
void GreenAtmosphere(char*	绿色氛围。
input, char* output)	, =, , ,
void GreenAtmosphere(char*	绿色氛围。支持 24 位 BMP 图像。
input, char* output, int	
width, int height, double a)	
void BMPtoJPG(char*	BMP 图像转为 JPG 图像。支持 24 位 BMP 图
input, char* output, int a)	像,尺寸必须是8的倍数,a代表文件压缩
	程度,数字越大,压缩后的文件体积越小,
	如 a=100。
void Mosaic(char*	马赛克化图像, input 是输入文件名,
input, char* output, int x)	output 是输出文件名。x 是马赛克处理的块
	的大小。支持 24 位 BMP 图像。
void MosaicFilter(char*	马赛克滤镜,参考: ratio=50。
input, char* output, int	
ratio)	
void MuddyAtmosphere(char*	泥色氛围。
input, char* output)	
void Expansion(char*	膨胀, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 4 位 BMP 图像。
void SmoothSharpen(char*	平滑, input 是输入文件名, output 是输出
input, char* output, int	文件名。Template 是平滑模板,均一化处
Template[3][3], int	理, coefficient1 = 9。支持24位BMP图
coefficient)	像。
void	高斯平滑, input 是输入文件名, output 是
GaussSmoothSharpen(char*	输出文件名。Template 是高斯平滑模板,
input, char* output, int	coefficient=16。支持 24 位 BMP 图像。

Template[3][3], int	
coefficient)	
void SobelSharpen(char*	Sobel 算子,input 是输入文件名,output
input, char* output, int	是输出文件名。Templatex 是 laplace 锐化
Templatex[3][3], int	模板,4 邻域,Templatey 是 laplace 锐化模
Templatey[3][3], int	板,8 邻域, coefficient1 = 9,
coefficient1, int	coefficient2 = 16。支持24位BMP图像。
coefficient2)	2.11
void MidSmoothing(char*	中值滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void AvgSmoothing(char*	均值滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void Averaging(char*	图像平均化, input 是输入文件名, output
input1, char* input2, char*	是输出文件名。a 是平均化相关参数,如
input3, char* output, int a)	a=3。支持 8 位 BMP 图像。
void PlaneSlicing(char*	平面切片, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void Translation(char*	图像平移,参考: xoffset=-100, yoffset=-
input, char* output, int	100°
xoffset, int yoffset)	
void	锐化空间滤波器, input 是输入文件名,
SharpeningSpatialFiltering8(output 是输出文件名。model 是锐化模板。
char* input, char* output, int	支持8位灰度图像。
mode1[9])	
void PseudoGrayscale(char*	伪灰度化, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void TwoColors(char*	二色化, input 是输入文件名, output 是输
input, char* output, int	出文件名。 threshold 是阈值,如
threshold, unsigned char	threshold=115; color1 和 color2 是要填
color1, unsigned char color2)	充的两个颜色。支持 24 位 BMP 图像。
void	filename 是生成的 PNG 图像文件名; img 是
PNGImageGeneration(char*	图像的像素数据,W 是图像的宽,H 是图像
filename, const unsigned char	的高,x=0 选择生成 RGB 图像,x=1 选择生
img[], unsigned W, unsigned	成 RGBA 图像。
H, int x) void MakeSphere(double	使用反射模型在正交投影下生成球体的图
void MakeSphere(double 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,	展图像文件名,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] =
The cond, char output)	{0.0, 0.0, 1.0}, S[3] = {0.0, 0.0,
	1.0}, r=50, a=0.5, m=1。支持 RAS 文件。
void MakeSphere(double	使用反射模型生成球体的图像, vector v
vector_v[3], double	是摄影机的方向,output 是输出的结果图
vector_s[3], double r, double	像文件名,ROWS 是输出图像的行数,COLS
, acapto i, double	M-> - 11 H / / C III H H H MH 11 3M, 0000

a, double m, int ROWS, int	是输出图像的列数,参考: vector_v[3] =
COLS, char* output)	$\{0.0, 0.0, 1.0\}, \text{ vector_s}[3] = \{0.0, 1.0\}$
	0.0, 1.0}, r=50, a=0.5, m=1。支持 RAS 文
	件。
void	双边滤波, input 是输入文件名, output 是
BilateralFiltering(string	输出文件名。支持 24 位 BMP 图像。ssd 和
input, char* output, double	sdid 分别是空间域标准差与强度域标准
ssd, double sdid)	差。
void	具有圆形结构集的双层形态侵蚀,支持8位
DoubleLayerErosion(char*	和 24 位 BMP 图像。
input, char* output)	
void SketchFilter(char*	素描滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void Zoom(char* input, char*	缩放,参考: scaleX=5, scaleY=5,
output, float scaleX, float	interpolation=0或interpolation=1。
scaleY, int interpolation)	
int Equal(char* input1, char*	若比对图像的梯度幅相似性偏差值等于 c
input2, double c)	则通过。input1 和 input2 是要比对的两个
	图像。c 是参考的阈值。支持 24 位 BMP 图
	像。
int GreaterThan(char*	若比对图像的梯度幅相似性偏差值大于 c
input1, char* input2, double	则通过。input1 和 input2 是要比对的两个
c)	图像。c 是参考的阈值。支持 24 位 BMP 图
	像。
int LessThan(char*	若比对图像的梯度幅相似性偏差值小于 c
input1, char* input2, double	则通过。input1 和 input2 是要比对的两个
c)	图像。c 是参考的阈值。支持 24 位 BMP 图
1 11 CMCD / 1	像。
double GMSD(char* input1,	求两幅图像的梯度幅相似性偏差值并返回
char* input2)	结果。input1和input2是要比对的两个图
	像。支持24位BMP图像。
<pre>void AddGaussNoise(char* input, char* output)</pre>	添加高斯噪声, input 是输入文件名,
void	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,
AddSaltPepperNoise(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void ChannelSeparation(char*	通道分离, input 是输入文件名, Routput
input, char* Routput, char*	是红色通道图像,Goutput 是绿色通道图
Goutput, char* Boutput)	像,Boutput 是绿色通道图像。支持 24 位
T. T	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。
void	图像无损解压,input 是待解压的文件名,
BMP24LosslessDecompression(c	output 是输出解压后的 BMP 文件名。支持
har* input, char* output)	24位BMP图像。
void	图像变色, input 是输入文件名, output 是

ImageDiscoloration(char*	输出文件名。如: a=0.2126, b=0.7152,
input, char* output, double	
a, double b, double c)	c=0.0722。支持 24 位 BMP 图像。
void HighPassFilter(char*	 高通滤波器,参考: preserve=0。
	同地怎伙品,多写: preserve-0。
input, char* output, int	
preserve)	泛服汗净品 乡老——————————————————————————————————
void EmbossFilter(char*	浮雕过滤器,参考: preserve=1。
input, char* output, int	
preserve)	<i>松</i>
void SharpenFilter(char*	锐化过滤器,参考: preserve=1。
input, char* output, int	
preserve)	业 1
void Convolution(char*	卷积,参考: w=7, preserve=1。
input, char* output, int w, int	
preserve)	는 바로 나바 씨마 스
void GaussianBlur(char*	高斯模糊,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	NE A ET III. A DE L
void HybridImage(char*	混合图像,参考: sigma=2, preserve=1。
input1, char* input2, char*	
output, float sigma, int	
preserve)	
void LowFrequencyImage(char*	低频图像,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage(char*	
input, char* output, float	
sigma, int preserve)	
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage1(char*	
input, char* output, float	
sigma, int preserve)	
void Bilateral(char*	双边滤波,参考: sigmal=3, sigma2=0.1。
input, char* output, float	
sigmal, float sigma2)	
void SkinSmooth(char*	皮肤细滑, a 是平滑级别, b 代表是否应用
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	

1 \	
h)	- Alt
void Shift(char* input, char*	Shift 函数,ch=1,v=0.1。
output, int ch, float v)	
void RGBtoHSV(char*	RGB 转 HSV。
input, char* output)	
void HSVtoRGB(char*	HSV 转 RGB。
input, char* output)	
void RGBtoLCH(char*	RGB 转 LCH。
input, char* output)	
void LCHtoRGB(char*	LCH 转 RGB。
input, char* output)	
void ImageCutting(char*	图像裁剪, input 是输入文件名, output 是
input, char* output, int	输出文件名。leftdownx,leftdowny,
leftdownx, int leftdowny, int	rightupx, rightupy 是要裁剪的矩形区域
rightupx, int rightupy)	的左下角和右上角的坐标(连续四个整数
	值,如 50 50 300 300)。支持 24 位 BMP 图
	像。
void	图像层算法。
ImageLayerAlgorithm(char*	
input, char* output)	
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, chai* 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)	
<pre>void PowerConvertion(char*</pre>	功率转换, input 是输入文件名, output 是
input, char* output, double	输出文件名。如: c = 1.2, g = 0.5。支持
c, double g)	8 位 BMP 图像。
void	拉普拉斯图像增强, input 是输入文件名,
LaplacianEnhancement(char*	output 是输出文件名。如: N=1。支持 8 位
input, char* output, int N, int	BMP 图像。
Lap1Mask[3][3])	参考模板:
	int Lap1Mask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void Smooth(char*	平滑, input 是输入文件名, output 是输出
input, char* output)	文件名。支持8位BMP图像。
void LaplaceSmooth(char*	拉普拉斯平滑, input 是输入文件名,
vora Lapraceomoctii (ciiai *	[J型日]型剂 I H, IIIput 是個八人门口,

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

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

```
-2, 0, 2,
                                              -1, 0, 1
                              彩色图以矩形方式局部截取且其它部分填
void
ColorRectangleLocalSegmentat
                              充,(x1,y1)是矩形的左上角的坐标,
ion(char*
                               (x2, y2) 是矩形的右下角的坐标。
                 input, char*
output, int
                              函数源代码:
             x1, int
                      y1, int
                              需引入以下头文件:
x2, int y2, BMPMat color)
                              typedef struct {
                                  unsigned char B;
                                  unsigned char G;
                                  unsigned char R;
                                  unsigned char A;
                              }BMPMat;
                              声明:
                              unsigned
                                          char**
                                                     BMPRead8(char*
                              input);
                              void
                                               GenerateImage8(char*
                              output, unsigned char** color);
                              BMPMat** BMPRead(char* input);
                              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 (idth; j++) {
                                        output[i][j].B=color.B;
                                        output[i][j]. G=color. G;
                                        output[i][j]. R=color. R;
```

```
for (unsigned
                                                           i
                                                   int
                                                                  =
                              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*
                 input, char*
                              (x2, y2) 是矩形的右下角的坐标。
                              函数源代码:
output, int
             x1, int
                      v1, int
                              需引入以下头文件:
x2, int
         y2, unsigned
                        char
color)
                              typedef struct {
                                  unsigned char B;
                                  unsigned char G;
                                  unsigned char R;
                                  unsigned char A;
                              }BMPMat;
                              声明:
                                                     BMPRead8(char*
                              unsigned
                                          char**
                              input);
                              void
                                               GenerateImage8(char*
                              output, unsigned char** color);
                              BMPMat** BMPRead(char* input);
                              void
                                                GenerateImage(char*
                              output, BMPMat** color, unsigned short
                              type);
                              unsigned int BMPHeight(char* input);
                              unsigned int BMPWidth(char* input);
                              参考例程:
                                  unsigned char color=255;
                                  unsigned
                                                             char**
                              input=BMPRead8(inputfile);
                                  unsigned
                                                             char**
                              output=BMPRead8(inputfile);
                                  unsigned
                                                                int
                              height=BMPHeight(inputfile);
```

```
unsigned
                                                                  int
                               width=BMPWidth(inputfile);
                                   for (unsigned
                                                     int
                                                             i
                               0; i < height; i++) {
                                       for (unsigned
                                                       int
                                                               j
                               0; j \le (idth; 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*
                  output, int
                               函数源代码:
input, char*
x1, int
           y1, int
                       x2, int
                               需引入以下头文件:
y2, BMPMat color)
                               typedef struct {
                                   unsigned char B;
                                   unsigned char G;
                                   unsigned char R;
                                   unsigned char A;
                               }BMPMat;
                               声明:
                               unsigned
                                                      BMPRead8(char*
                                           char**
                               input);
                               void
                                                GenerateImage8(char*
                               output, unsigned char** color);
                               BMPMat** BMPRead(char* input);
                               void
                                                 GenerateImage(char*
                               output, BMPMat** color, unsigned short
                               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 \leq 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 \leq height; i++) {
         for (unsigned
                           int
0; j \le (idth; j++) {
            if(j)=x1&&j<=x2&&i==y1)
             output[i][j]. B=color. B;
           output[i][j]. G=color. G;
           output[i][j]. R=color. R;
             if(j==x1\&\&i>=y1\&\&i<=y2)
             output[i][j]. B=color. B;
           output[i][j]. G=color. G;
           output[i][j]. R=color. R;
             if(j==x2\&\&i>=y1\&\&i<=y2)
             output[i][j].B=color.B;
           output[i][j]. G=color. G;
           output[i][j].R=color.R;
             if(j)=x1&&j <=x2&&i==y2
             output[i][j]. B=color. B;
           output[i][j]. G=color. G;
           output[i][j]. R=color. R;
```

```
GenerateImage (outputfile, output, 24);
                               灰度图画矩形,(x1, y1)是矩形的左上角
void GrayDrawRectangle(char*
                               的坐标,(x2, y2)是矩形的右下角的坐标。
input, char*
                  output, int
                               函数源代码:
x1, int
           y1, int
                       x2, int
y2, unsigned char color)
                               需引入以下头文件:
                               typedef struct {
                                   unsigned char B;
                                  unsigned char G;
                                  unsigned char R;
                                  unsigned char A;
                               }BMPMat;
                               声明:
                               unsigned
                                           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:
                                   unsigned char color=255;
                                  unsigned
                                                               char**
                               input=BMPRead8(inputfile);
                                  unsigned
                                                               char**
                               output=BMPRead8(inputfile);
                                  unsigned
                                                                  int
                               height=BMPHeight(inputfile);
                                  unsigned
                                                                  int
                               width=BMPWidth(inputfile);
                                   for (unsigned
                                                     int
                                                             i
                               0; i \leq height; i++)
                                       for (unsigned
                                                       int
                                                              j
                               0; j \le dth; j++) {
                                         output[i][j]=color;
                                   for (unsigned
                                                     int
                                                             i
                               0; i \leq height; i++)
                                       for (unsigned
                                                       int
                                                              j
                               0; j \le (idth; j++) {
```

```
if(j)=x1\&\&j<=x2\&\&i==y1)
                                           output[i][j]=color;
                                           if(j==x1\&\&i>=y1\&\&i<=y2)
                                           output[i][j]=color;
                                           if(j==x2\&\&i>=y1\&\&i<=y2)
                                           output[i][j]=color;
                                           if(j)=x1\&\&j<=x2\&\&i==y2)
                                           output[i][j]=color;
                                  }
                              GenerateImage8(outputfile, output);
                              X方向翻转,支持 JPG 文件。
void flipX(char* input, char*
output)
void flipY(char* input, char*
                              Y方向翻转,支持 JPG 文件。
output)
                              裁剪。
void Crop(char* input, char*
output, uint16_t
                    start_x,
uint16 t start y,
                    uint16 t
new_height,
                    uint16_t
new width)
                Resize(char*
                              缩放。
void
                  output, int
input, char*
new width, int new height)
void Scale(char* input, char*
                              比例。
output, double ratio)
                               灰度平均值。
void
          GrayscaleAvg(char*
input, char* output)
void
          grayscaleLum(char*
                               灰度亮度。
input, char* output)
                               彩色遮罩。
             ColorMask(char*
void
input, char*
                output, float
r, float g, float b)
void
              PixeLize(char*
                              像素化,参考: strength=2。
input, char*
                  output, int
strength)
```

void GaussianBlur(char*	高斯模糊,参考: strength=2。
input, char* output, int	同分所交份,多一号· Strength 2。
strength)	
void EdgeDetection(char*	 边缘检测,参考: cutoff=115。
input, char* output, double	之外证例,多与. cutoff file。
cutoff)	
void Sharpen(char*	锐化。
input, char* output)	见心。
void GrayAVS (char*	input 是输入文件名, output 是输出文件
input, char* output, float	Tilput 定栅八叉行石, output 定栅凸叉行
k, float b)	石。又可可见MI 图像。
void	古古凤均衡化 input 見給) 立 併 夕
	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)	一片儿
void Binarization(char*	二值化。
input, char* output)	가축마#AZW
void	分离出蓝色通道。
ChannelSeparation_B(char*	
input, char* output)	11 1. 17 A 17 W
void	分离出绿色通道。
ChannelSeparation_G(char*	
input, char* output)	
void	分离出红色通道。
ChannelSeparation_R(char*	
input, char* output)	
void Inverse(char*	反转。
input, char* output)	
void	直方图均衡化。
HistogramEqualization8(char*	
input, char* output)	
void Smooth(char*	平滑。
input, char* output)	
void CannyEdge(char*	Canny 算子。
input, char* output)	
void EdgeEnhance(char*	边缘增强。
input, char* output)	
void AvrFilter(char*	input 是输入文件名, output 是输出文件
input, char* output1, char*	名。如 M=21, N=1。支持 8 位 BMP 图像。
output2, int M, int N)	
void GryOppositionSSE(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
input, ondi	1。次11。压2 口以。

void MedianFilter(char*	中值滤波器, input 是输入文件名, output
input, char* output, int M, int	是输出文件名。如 M=5, N=5。支持 8 位 BMP
N)	图像。
void EdgeSharpeningGry(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
void SJGryandRiceTest(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
void TextTest(char*	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
void RedChannel(char*	生成图像的红色通道图像, input 是输入文
input, char* output)	件名,output 是输出文件名。支持 24 位 BMP
	图像。
void GreenChannel(char*	生成图像的绿色通道图像, input 是输入文
input, char* output)	件名,output 是输出文件名。支持 24 位 BMP
	图像。
void BlueChannel(char*	生成图像的蓝色通道图像, input 是输入文
input, char* output)	件名,output 是输出文件名。支持 24 位 BMP
	图像。
void	直方图统计, input 是输入文件名, output
HistogramStatistics(char*	是输出文件名。支持 24 位 BMP 图像。
input, char* output)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization1(char*	output 是输出文件名。支持 24 位 BMP 图
input, char* output)	像。
void ReflectionRay(char*	反射线, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void MeanFiltering24(char*	均值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void MedianFiltering24(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void ZoomOutAndZoomIn(char*	缩放(双线性插值), input 是输入文件名,
input, char* output, double	output 是输出文件名。value 是放大倍数,
value)	如 value=0.5。支持 24 位 BMP 图像。
void Translation24(char*	平移, input 是输入文件名, output 是输出
input, char* output, int x, int	文件名。x 是横轴的平移量,y 是纵轴的平
y)	移量,如 x=-10, y=-30。支持 24 位 BMP 图
	像。
void Mirror24(char*	镜像, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 24 位 BMP 图像。
void Rotate24(char*	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)	threshold 是给定的阈值,如 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。
	1 112
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)	
void Repudiation(char*	将伪彩图片反白, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 24 位 BMP 图
	像。
void Grayl(char* input, char*	将彩色图片变成灰度图片, input 是输入文
output)	件名,output 是输出文件名。支持 24 位 BMP
outputy	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
	1
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
	}Image1;
void SaveImage1(char*	将 Image1 型数据保存为 BMP 图像,output
output, Image1* img)	是生成的 BMP 图像文件名, img 是要保存的
	图像数据。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
<u>L</u>	

	unsigned char* Data; //像素数据
	}Image1;
void	图像对比度扩展, input 是输入文件名,
ImageContrastExtension(char*	output 是输出文件名。
input, char* output, double	其中,可参考: double
m, double g1, double g2, double	m=1.5, g1=100.0, g2=200.0; m 对应斜率
a)	double $a=(255.0-m*(g2-g1))/(255.0-$
	(g2-g1));
	支持8位BMP图像。
void Binaryzation(char*	图像二值化,input 是输入文件名,output
input, char* output, int	是输出文件名。threshold 是将灰度值转化
threshold)	为二值的阈值,如 threshold=80。支持 24
	位 BMP 图像。
void	全局二值化,input 是输入文件名,output
GlobalBinarization(char*	是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	自适应二值化, input 是输入文件名,
AdaptiveBinarization(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	膨胀操作, input 是输入文件名, output 是
ExpansionOperation(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	腐蚀操作, input 是输入文件名, output 是
CorrosionOperation(char*	输出文件名。支持 8 位 BMP 图像。
input, char* output)	7 11 1/2 · 12 14 \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
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 Negative1(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 T int border)	阈值,border 是边界范围,如: T=50, border=0。
T, int border)	缩放图片(最近邻插值法)。
	湘从闰月(取处邻州徂伝)。
TransformShapeNearest (char*	
<pre>inputFile, char* output, unsigned int newWidth,</pre>	
unsigned int newHeight)	

	始 对因止(四州州括估)
void	缩放图片(双线性插值法)。
TransformShapeLinear(char*	
inputFile, char* output,	
unsigned int newWidth,	
unsigned int newHeight)	图换的任亲 免庇的按结
void	图像的任意角度的旋转。
TransformShapeWhirl(char*	
inputFile, char* output,	
float angle)	圆角的总格 翻法
void	图像的镜像翻转。
TransformShapeUpturn(char*	
inputFile, char* output, int	
a)	业务图妹友库图 对工 C1-W-1- 的
void TransformColorCroygools(char	彩色图转灰度图,对于 GrayscaleMode 的
TransformColorGrayscale(char	值: 1表示加权法, 2表示最值法, 3表示
* inputFile, char* output, int	均值法,4表示红色分量法,5表示绿色分
GrayscaleMode)	量法,6表示蓝色分量法。
void	二值图(大津法 OSTU,适用双峰直方图。)
TransformColorBWOSTU(char*	
inputFile, char* output)	
void	二值图(三角法 TRIANGLE,适用单峰直方
TransformColorBWTRIANGLE(cha	图。)
r* inputFile, char* output)	
void	二值图(自适应阈值法,areaSize=25 较合
TransformColorBWAdaptive(cha	适)
r* inputFile, char* output,	
int areaSize)	
void	二值图(用二值图表示灰度变
TransformColorBWGrayscale(ch	化, areaSize=25 较合适)
ar* inputFile, char* output,	
int areaSize)	三
void	反色。
TransformColorOpposite(char*	
inputFile, char* output)	古子园均临4/八上7/9 数甲再加圣红
void	直方图均衡化(分步计算,效果更加柔和)。
TransformColorHistogramPart(
char* inputFile, char*	
output)	古子园均临4/1 数4/1 数 英甲亚基小型
void	直方图均衡化(整体计算,效果更加尖锐)。
TransformColorHistogramAll(c	
har* inputFile, char* output)	坐付担<i>作</i> (カウツ)
void KernelsUseDIY(char*	巻积操作(自定义)。
inputFile, char* output,	
double* kernels, int	

areaSize, double modulus)	
void	中值滤波。
WavefilteringMedian(char*	
inputFile, char* output)	
void	高斯滤波。a=9, b=1.0/16。
	高斯滤波卷积核:
WavefilteringGauss(char*	
inputFile, char*	double KERNELS_Wave_Gauss[9] =
output, double*	{
KERNELS_Wave_Gauss, int	1, 2, 1,
a, double b)	2, 4, 2,
	1, 2,1
	};
void	低通滤波。areaSize=9,modulus=1。
Wavefiltering_LowPass(char*	// 低通滤波卷积核 LP1
inputFile, char* output,	double KERNELS_Wave_LowPass_LP1[9] =
double* kernels, int	{
areaSize, double modulus)	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
	};
void	高通滤波。areaSize=9, modulus=1。
WavefilteringHighPass(char*	
inputFile, char* output,	double KERNELS_Wave_HighPass_HP1[9] =
double* kernels, int	{
areaSize, double modulus)	-1, -1, -1,
areastre, doubte modulus/	-1, 9, -1,
	-1, -1, -1
	}.
	J,

```
// 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
void
                             均值滤波。areaSize=25, modulus=1.0 /
Wavefiltering Average(char*
                             25。
                             // 均值滤波卷积核
inputFile, char*
                             double KERNELS Wave Average[25] =
output, double*
KERNELS Wave Average, int
areaSize, double modulus)
                               1, 1, 1, 1, 1,
                               1, 1, 1, 1, 1,
                              1, 1, 1, 1, 1,
                              1, 1, 1, 1, 1,
                               1, 1, 1, 1
void
                             差分边缘检测。areaSize=9, modulus=1。
                             // 差分垂直边缘检测卷积核
EdgeDetectionDifference(char
* inputFile, char* output,
                             double
                kernels, int
double*
                             KERNELS_Edge_difference_vertical[9] =
areaSize, double modulus)
                                0, 0, 0,
                                -1, 1, 0,
                                0, 0, 0
                             };
                             // 差分水平边缘检测卷积核
                             doub1e
                             KERNELS_Edge_difference_horizontal[9]
                                0, -1, 0,
                                0, 1, 0,
```

```
0, 0, 0
                             };
                             // 差分垂直和水平边缘检测卷积核
                             double KERNELS_Edge_difference_VH[9]
                                -1, 0, 0,
                                0, 1, 0,
                                0, 0, 0
                             Sobel 边缘检测。
void
KernelsUseEdgeSobel(char*
                             // Sobel X 边缘检测卷积核
            char*
inputFile,
                    output,
                             double KERNELS_Edge_Sobe1_X[9] =
double*
         kernels1,
                    double*
                                -1, 0, 1,
kernels2)
                                - 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
void
                             Laplace 边缘检测。areaSize=9,
                             modulus=1.
EdgeDetectionLaplace(char*
                             // Laplace 边缘检测卷积核 LAP1
inputFile, char*
                             double KERNELS_Edge_Laplace_LAP1[9] =
output, double*
                kernels, int
areaSize, double modulus)
                                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
void MorphologyErosion(char*
                           腐蚀。
                           // 腐蚀卷积核
inputFile, char*
output, double* kernels)
                           doub1e
                           KERNELS Morphology Erosion cross[9] =
                               0, 1, 0,
                               1, 1, 1,
                               0, 1, 0
                            };
                            膨胀。
void
MorphologyDilation(char*
                           // 膨胀卷积核
inputFile, char*
                           doub1e
output, double* kernels)
                           KERNELS_Morphology_Dilation_cross[9]
                               0, 1, 0,
                               1, 1, 1,
                               0, 1, 0
                            人脸检测。Threshold 是二值化时的阈值,
void
        FaceDetection(char*
                           areaSize1=25, modulus1=1.0/25, 对于
input, char*
             output, double*
KERNELS_Wave_Average, int
                           GrayscaleMode 的值: 1表示加权法,2表
GrayscaleMode, int
                           示最值法,3表示均值法,4表示红色分量
                           法,5表示绿色分量法,6表示蓝色分量法;
areaSize, unsigned
                      char
                           GrayscaleMode=1, areaSize=25, B=0, G=0,
Threshold, unsigned
                      char
B, unsigned char G, unsigned
                           R=200, a=1 表示选择均值滤波预处理图像,
char
                           a=2 表示选择高通滤波预处理图像, a=3 表
      R, int
              a, int
                     b, int
                           示选择低通滤波预处理图像, a=4 表示选择
areaSizel, double modulus1)
```

	高斯滤波预处理图像, a=5 表示选择中值滤
	波预处理图像, b=1 表示自适应阈值法, b=2
	表示自定义阈值法,b=3表示大津法 OSTU,
	b=4 表示三角法 TRIANGLE, b=5 表示一般方
	法。
void ImageEncryption(char*	图像加密,支持8位、24位和32位BMP图
inFileName, char∗	像。inFileName 是原图图像文件名,
outFileName, char key)	outFileName 是解密图像文件名,key 是密
	钥,如 key=255。
void ImageDecryption(char*	图像解密, in File Name 是加密图像文件名,
inFileName, char*	outFileName 是解密图像文件名,key 是密
outFileName, char key)	钥,如 key=255。支持 8 位、24 位和 32 位
,	BMP 图像。
void	图像加解密, Key 是密钥, a=1 时执行加密,
EncryptionDecryption(char*	a=0 时执行解密。支持 24 位 BMP 图像。
input, char* output, int	The state of the s
Key, int a)	
void Encryption(char*	图像加密, input 是输入文件名, output 是
input, char* output, int Key)	输出文件名。Key 是密钥。支持 24 位 BMP 图
Impac, char - daupac, inc hey)	像。
void Decryption(char*	图像解密, input 是输入文件名, output 是
input, char* output, int Key)	输出文件名。Key 是密钥。支持 24 位 BMP 图
input, char* output, int key)	他。
void Compress8(string	图像压缩, input 是输入文件名, output 是
	输出文件名。支持 8 位 BMP 图像。
input, string output) void Decompression(string	图像解压, input 是输入文件名, output 是
1 , 0	
input, string output)	输出文件名。支持 8 位 BMP 图像压缩后的结
	果文件。
void HorizontalMirror(char*	水平镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void MirrorVertically(char*	垂直镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void XMirroring(char*	X 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void YMirroring(char*	Y镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void ImageConvolution(char*	图像卷积, input 是输入文件名, output 是
input, char* output, double**	输出文件名。Kernel 是卷积核,如 double
Kernel, int n, int m)	Kernel[3][3] = {{-0.225, -0.225-
	0 00=) (0 00= : 0 00=) (
1	0. 225}, {-0. 225, 1, -0. 225}, {-0. 225, -
	0.225, -0.225}}; n 是 Kernel 的第一维的
	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
void SpatialMeanFiter(char*	0.225, -0.225}}; n 是 Kernel 的第一维的

input, char* output, int	
radius)	2.20 1 Max 1 2 Mars - 22 Max - 23 Max -
void	空间中值过滤器,参考: radius=3。
SpatialMedianFiter(char*	
input, char* output, int	
radius)	
void SpatialMaxFiter(char*	空间最大过滤器,参考: radius=3。
input, char* output, int	
radius)	
void SpatialMinFiter(char*	空间最小过滤器,参考: radius=3。
input, char* output, int	
radius)	
void SpatialGaussFiter(char*	空间高斯过滤器,参考: radius=3。
input, char* output, int	
radius)	
void	空间统计滤波器,参考: radius=3, T=0.2。
SpatialStatisticalFiter(char	
* input, char* output, int	
radius, float T)	
void Mosaic(char*	马赛克滤镜, w 和 h 是输出图像的宽和高。
input, char* output, int w, int	支持 PNG 图像。
h)	
void FFTAmp(char*	FFT 放大器,参考: inv=false。
input, char* output, bool inv)	
void FFTPhase(char*	FFT 相位,参考: inv=false。
input, char* output, bool inv)	
void STDFT1(char*	参考: inv=false。
input, char* output, bool inv)	
void STDFT2(char*	参考: inv=false。
input, char* output, bool inv)	
void SpectrumShaping(char*	图像频域滤波, FFT 变换相位谱,
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 Nesting(char*	图像嵌套, Biginput 是输入的大图,
Biginput, char*	Smallinput 是输入的小图。支持 24 位 BMP
Smallinput, char* output)	图像。
void Blend(char*	图像融合之混合化, input1 和 input2 是输
input1, char* input2, char*	入的两个要融合的图像, output 是输出文
output)	件名。支持 24 位 BMP 图像。
	· ·

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

output, float ExpScalValue)	
void	最近邻插值, input 是输入文件名, output
NearestNeighborInterpolation	是输出文件名。ExpScalValue 是期望的缩
Scaling(char* input, char*	放倍数(允许小数)。支持 BMP 图像。
output, float ExpScalValue)	7 - 7 - 7 - 7 - 7 - 7 - 7 - 7 - 7 - 7 -
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)	
vector < BMPMat >	返回图像的所有颜色种类。
FXH_CompressionColor(BMPMat*	
* input, int width, int height)	
vector <vector<coor></vector<coor>	返回各个颜色种类对应的所有坐标值,
FXH_CompressionCoor(BMPMat**	FXH_CompressionColor 函数与
input, int width, int height)	FXH_CompressionCoor 函数的返回数据按
	顺序一一对应。
BMPMat**	图像复原, width 是原图像的宽, height 是
ImageRestoration(vector <bmpm< td=""><td>原图像的高。</td></bmpm<>	原图像的高。
at>	
color, vector <vector<coor></vector<coor>	
coor, int width, int height)	
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)	图确选注 innut 目於) 文件名 autaut 目
void ImageRotation(char*	图像旋转, input 是输入文件名, output 是输出文件名。支持 8 位 BMP 图像。angle 是
input, char* output, double angle)	棚山文件石。文符 o 位 DMF 图像。alig1e 定 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	窗像旋转,Input 走棚八叉件石,output 走 输出文件名。支持 8 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标, color 是旋转后非原
color)	图区域的填充颜色。
void Rotation24(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 24 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标; red、green、blue 分
red, unsigned char	别是旋转后非原图区域要填充的颜色的红
green, unsigned char blue)	绿蓝分量。
void Rotation(char*	图像旋转, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。angle 是
angle, unsigned char color)	旋转的角度, color 是旋转后非原图区域填

	充的颜色,如 color=100。
void Rotate(char*	图像旋转, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 BMP 图像。angle 是旋转
angle)	的角度。
void NoiseUniform(char*	均匀分布噪声,参考: a=0, b=0.2。
· ·	均匀为和噪户,参考: a=0,b=0.2。
input, char* output, double	
a, double b)	· 古世吧士 - 分型
void NoiseGauss(char*	高斯噪声,参考: mean=0, delta=31。
input, char* output, float	
mean, float delta)	世纪四 士
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 m1, int a, int b)	第一维的大小, m1 是第二维的大小, 形如
	str[n][m1]; a 和 b 是相关的调节参数,可
	以为 a=3, b=5。
	参考模板:
	$char str[6][8] = \{ \{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, $
	0, 0, }, { 255, 0, 255, 0, 0, 255, 0,
	0 },
	{ 255, 0, 255, 255, 0, 255, 0,
	255 }, { 255, 255, 255, 0, 0, 255,
	255, 255 },
	{ 255, 0, 255, 255, 0, 255, 255,
	255 }, { 0, 255, 255, 255, 255, 255,
	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)	H1回冰天日口。天月 0 四/H 2 H 匹 四/M 图像。
int	 返回图像像素的最大值,filename 是输入
MaximumValueOfImagePixels(ch	的图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	15日周梅梅末始45世 0·1 日秋×4
float	返回图像像素的均值,filename 是输入的
AverageValueOfImagePixels(ch	图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	

double	返回图像像素的标准差,filename 是输入
StandardDeviationOfImagePixe	的图像文件名。支持 8 位和 24 位 BMP 图像。
ls(char* filename)	
double EntropyOfImage(char*	返回图像的熵,支持8位和24位BMP图像。
filename)	
float*	filename 是输入的图像文件名。存储每个
CountTheFrequencyOfPixels(ch	像素的频率,像素值为0~255,返回值数组
ar* filename)	中的元素序号即为像素值,该序号在数组下
	的值即为这个像素的频率。支持 8 位和 24
	位 BMP 图像。
void Rotate(char*	图 像 旋 转 。 参 考 : angle=80 ,
input, char* output, int	interpolation=0或interpolation=1。
angle, int interpolation)	
void HSV(char* input, char*	图像色调饱和度明度调节,参考: h=120,
output, int h, int s, int v)	s=60, v=20。
void OilpaintFilter(char*	油画滤镜,参考: radius=10, smooth=100。
input, char* output, int	
radius, int smooth)	
unsigned int*	圆检测,返回圆心的坐标和半径,返回值数
CircleDetection1(char*	组中第一个元素是圆心 X 坐标,第二个元素
input)	是圆心 Y 坐标,第三个元素是圆半径。
VECTORS	圆检测,返回圆心的坐标和半径,返回以圆 10. X 44 5
CircleDetection2(char* input)	心 X 坐标,圆心 Y 坐标,圆半径组成的结构 体。
	冲。 需引入以下结构体:
	typedef struct {
	int* x;
	int* y;
	unsigned int size;
	} VECTORS;
SWARM*	圆检测,返回圆心的坐标和半径,返回圆检
CircleDetection3(char*	测结果。
input)	需引入以下结构体:
	typedef struct
	{
	float* Xi; // Posicion
	float* Vi; // Velocidad
	float* Pi; // Mejor posicion
	float XFit; // Posición fitness
	float PFit; // Mejor posición
	fitness
	} PARTICLE;
	typedef struct
	<u> </u>

	unsigned int nParticles;// Número
	de PARTICLES del SWARM
	unsigned int nParams;
	unsigned int idGbest; // indice a
	la mejor partícula
	float c1; //Las constantes
	dan peso a la experiencia
	float c2; //indivudual o a
	la colectiva
	float Vmin;
	float Vmax;
	PARTICLE* Swarm; //Apuntador a SWARM
	SWARM;
vector <vector<float> ></vector<float>	圆检测,返回圆心的坐标和半径,返回圆检
CircleDetection4(char*	测结果。
input)	MISH VV 0
void HaloFilter(char*	 晕角滤镜,参考: ratio=100。
,	半刀1心克,少~5:1at110=100。
ratio)	大安古子园 分类 1W:1/1 0FC
void GrayHistogram(char*	灰度直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void RedHistogram(char*	红色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void GreenHistogram(char*	绿色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void BlueHistogram(char*	蓝色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization2(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output, int	BMP 图像。imgBit 是输入图像的位数。
imgBit)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualization3(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output)	BMP 图像。
void	直方图均衡化, input 是输入文件名,
HistogramEqualization4(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output)	BMP 图像。input 是输入文件名称, out 是
input, onar. output/	输出文件名称。
void	直方图均衡化,参考: hWidth=256,
	且为图均衡化,参考: nwidth=250, hHeight=100。
HistogramEqualization(char*	Interdir—100°

input, char* output, int	
hWidth, int hHeight)	1 P + 1 F
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 MedianFiltering1(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void MedianFiltering2(char*	中值滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
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 Thresho1d=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)	
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 *
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,
void WhiteBalance(const	FILE_TYPE_AD=(0x4441), a=0x80, b=1。 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*
input, char* output, float
probability)

噪声化图像, probability=0.1。支持 PGM 和 PBM 图像。

void

HoughTransformCircle1(char*
input, char* output, 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。

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

void ShapeEdgeDetection1(char* input,char* output,unsigned	gamma==1.0。 返回以(vCenter,hCenter)和半径 (vRadius,hRadius)为中心的椭圆数据, 共有 number 组数据,每组包含一个椭圆数 据,第一个元素是 vCenter,第二个元素是 hCenter,第三个元素是 vRadius,第四个元 素是 hRadius。 支持 PGM 和 PBM 图像。 形状边缘检测, CANNY_THRESH4=35, CANNY_blur4=7。支持 PNG 图像。
char CANNY_THRESH4, int	
canny_blur4) void ShapeEdgeDetection2(char* input, char* output, unsigned char CANNY_THRESH4, int CANNY blur4)	形状边缘检测,CANNY_THRESH4=35,CANNY_blur4=7。支持PNG图像。
void ShapeEdgeDetection3(char* input, char* output, unsigned char CANNY_THRESH, int CANNY_BLUR)	形 状 边 缘 检 测 , CANNY_THRESH=50 , CANNY_BLUR=12。支持 PNG 图像。
void ShapeEdgeDetection4(char* input,char* output,unsigned char CANNY_THRESH,int CANNY_BLUR)	形 状 边 缘 检 测 , CANNY_THRESH=50 , CANNY_BLUR=12。支持 PNG 图像。
void ShapeEdgeDetection5(char* input,char* output,unsigned char CANNY_THRESH2,int CANNY_BLUR2)	形状边缘检测,CANNY_THRESH2=10, CANNY_BLUR2=2。支持PNG图像。
void ShapeEdgeDetection6(char* input,char* output,unsigned char CANNY_THRESH2,int CANNY_BLUR2)	形状边缘检测,CANNY_THRESH2=10, CANNY_BLUR2=2。支持PNG图像。
void ShapeEdgeDetection7(char* input, char* output, unsigned char CANNY_THRESH3, int CANNY_blur3)	形状边缘检测,CANNY_THRESH3=45,CANNY_blur3=10。支持PNG图像。
void	形状边缘检测,CANNY_THRESH3=45,

ShapeEdgeDetection8(char*	CANNY_blur3=10。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH3, int	
CANNY_blur3)	

其他处理

<pre>void DES_Encrypt(char *PlainFile,</pre>	DES 加密函数,支持多种文件。
char *Key,char *CipherFile)	PlainFile 是原文件的文件名,Key 是
	密钥字符,CipherFile 是加密后的文
	件名。
<pre>void DES_Decrypt(char *CipherFile,</pre>	DES 解密函数,支持多种文件。
char *Key,char *PlainFile)	CipherFile 是已加密文件的文件名,
	Key 是密钥字符,PlainFile 是解密后
	的文件名。
<pre>uint8_t* AESencrypt (uint8_t*</pre>	AES 加密函数,input 是原数据,key
input, uint8_t* key, int size)	是密钥,size 是 input 的大小。返回
	加密结果数据。
<pre>uint8_t* AESdecrypt (uint8_t*</pre>	AES 解密函数,input 是已加密数据,
input, uint8_t* key, int size)	key 是密钥,size 是 input 的大小。
	返回解密结果数据。
void Encode(char* input, char*	文本文件压缩, input 是输入文件名,
output)	output 是输出文件名。
void Decode(char* input, char*	文本文件压缩结果解压缩, input 是
output)	输入文件名,output 是输出文件名。
void FileCompress(char *input ,	文件压缩, input 是输入文件名,
char *output)	output 是输出文件名。
void FileDecompression(char	文件压缩结果解压缩, input 是输入
*input , char *output)	文件名,output 是输出文件名。

高级算子

int* TemplateMatch(char*	模板匹配, input 是母本图像,
input, char* Template, char*	Template 是样本图像,output 是结果
output, int channels, int ROTATION)	图像文件名。channels 是图像的像素
	通道数,返回值数组中的第一个元素
	是最大匹配分数,第二个元素是目标
	的 X 轴坐标值,第三个元素是目标的 Y
	轴坐标值。至少支持 PNG 图像。参考:
	channels=3 , ROTATION=360 或
	ROTATION=1。
vector <sol> TemplateMatch1(char*</sol>	模板匹配, input 是母本图像,
input, char* Template, char*	Template 是样本图像,output 是结果
output, int channels, int ROTATION)	图像文件名。channels 是图像的像素
	通道数,返回匹配结果数组。至少支持
	PNG 图像。参考: channels=3,

	ROTATION=360 或 ROTATION=1。
	需引入以下结构体:
	typedef struct SumSquareD {
	int SSDr, SSDg, SSDb, sum;
	SSD;
	<pre>typedef struct solutionLocation {</pre>
	int x, y;
	SSD colorD;
	} sol;
int* TemplateMatch(char*	模板匹配, input 是母本图像,
input, char* Template, int	Template 是样本图像。channels 是图
channels, int ROTATION)	像的像素通道数,返回值数组中的第
,	一个元素是最大匹配分数,第二个元
	素是目标的 X 轴坐标值,第三个元素
	是目标的Y轴坐标值。至少支持PNG图
	像。参考: channels=3, ROTATION=360
	或 ROTATION=1。
vector <sol> TemplateMatch1(char*</sol>	模板匹配, input 是母本图像,
input, char* Template, int	Template 是样本图像,output 是结果
channels, int ROTATION)	图像文件名。channels 是图像的像素
	通道数,返回匹配结果数组。至少支持
	PNG 图像。参考: channels=3,
	ROTATION=360 或 ROTATION=1。
	需引入以下结构体:
	typedef struct SumSquareD {
	int SSDr, SSDg, SSDb, sum;
	} SSD;
	<pre>typedef struct solutionLocation {</pre>
	int x, y;
	SSD colorD;
	} sol;
void PGMSobel(char* input,char*	Sobel 算子, input 是输入文件名,
output, int Mx[3][3], int	output 是输出文件名。支持 P5 格式的
My[3][3], int max, int min)	PGM 文件。
	参考模板:
	int $Mx[3][3] = \{\{-1, 0, 1\}, \{-2, \}\}$
	$0, 2$, $\{-1, 0, 1\}$
	int $My[3][3] = \{\{-1, -2, -1\}, \{0, \}\}$
	$0, 0$, $\{1, 2, 1\}$
	int max = -9999
	int min = 9999
void PGMSobelX(char* input, char*	X 方向滤波, input 是输入文件名,
output, int Mx[3][3], int	output 是输出文件名。支持 P5 格式的
My[3][3], int max, int min)	PGM 文件。

_	
<pre>void PGMSobelY(char* input, char* output, int</pre>	参考模板: int Mx[3][3] = {{-1, 0, 1}, {-2, 0, 2}, {-1, 0, 1}} int My[3][3] = {{-1, -2, -1}, {0, 0, 0}, {1, 2, 1}} int max = -9999 int min = 9999 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 文件。 参考模板: int Mx[3][3] = {{-1, 0, 1}, {-2, 0, 2}, {-1, 0, 1}} int My[3][3] = {{-1, -2, -1}, {0, 0, 0}, {1, 2, 1}} int max = -9999 int min = 9999 Sobel 算子, input 是输入文件名,
<pre>void PGMSobell(char* input, char* output, int min, int max, int mx[3][3], int my[3][3])</pre>	Sobel 异子, Input 是期人文件名, output 是输出文件名。min 和 max 是图像归一化相关参数,如 min = 1000000, max = 0; mx 和 my 分别是Sobel 算子的 X 和 Y 方向模板。支持P2 和 P5 格式的 PGM。参考模板: int mx[3][3] = {
<pre>void PGMSobelX1(char* input, char* output, int min, int max, int mx[3][3], int my[3][3])</pre>	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 = 1000000, max = 0; mx 和 my 分别是 Sobel 算子的 X 和 Y 方向模板。支持 P2 和 P5 格式的 PGM。 参考模板: int mx[3][3] = {

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

<pre>void VertSobel(char* input, char* output)</pre>	垂直 Sobel 算子, input 是输入文件 名, output 是输出文件名。支持 P5 格 式的 PGM 图像。
<pre>void PGMSobell(char* input, char* output, int threshold)</pre>	Sobel 算子, input 是输入文件名, output 是输出文件名。支持 P5 格式的 PGM 图像。threshold 是目标阈值, 如 threshold=80。
<pre>void YFiltering(char* input, char* output, int</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},
	$\{0, -2\},\ \{0, -1\}\};$
	int sobel_y[3][3] = { { 1, 2, 1},
	0, 0}, -2, -1}};
<pre>void XFiltering(char* input, char* output, int</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, } , } , }
	$\{0, -2\},\ \{0, -1\}\};$
	int sobel_y[3][3] = { { 1, } },
	{ 0, 0}, {-1,
0.1.10.1	-2, -1}};
<pre>void SobelFiltering(char* input, char* output, int sobel_x[3][3], int sobel_y[3][3])</pre>	Sobel 算子, input 是输入文件名, output 是输出文件名。支持 P5 格式的 PGM 图像。
	参考模板: int_sobel_x[3][3] = {

	_11
	-1},
	{ 2,
	[0, -2],
	{ 1,
	$\{0, -1\}\};$
	int sobel_y[3][3] = { { 1,
	2, 1},
	{ 0,
	$\{0, 0\},$
	{-1,
	$-2, -1\}\};$
D 1. D 1.	
void PrewittFiltering(char*	Prewitt 算子, input 是输入文件名,
input, char* output, int	output 是输出文件名。支持 P5 格式的
prewitt x[3][3], int	PGM 图像。
prewitt_y[3][3])	参考模板:
prowrete_y [o] [o])	
	int prewitt_x[3][3] = { { 5, 5,
	5},
	$\{ -3, $
	[0, -3],
	{ -3, -
	·
	3, -3}};
	$int prewitt_y[3][3] = \{ \{ 5, \} \}$
	$-3, -3\},$
	{ 5, 0,
	-3},
	{5, -3,
	-3}};
void LaplacianFiltering(char*	Laplace 算子,input 是输入文件名,
input, char* output, int	output 是输出文件名。支持 P5 格式的
laplacian[3][3])	PGM 图像。laplacian 是 Laplacian 算
	子模板。
	参考模板:
	int laplacian[3][3] = { { 1, 1,
	1},
	{ 1, -
	8, 1},
	{ 1,
	1, 1}};
vector <float></float>	支持 P5 和 P6 格式的 PPM 文件, 从返
HarrisCornerDetection(char*	回值数组中的第一个元素开始,返回
input, int width, int height, int	值以3个元素为一组,分别是角点的X
channels, int step, float	坐标、Y 坐标和分数,若返回值数组名
threshold, float k, float sigma)	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
	WH4 >> 4H / WEG J, ME 1 J, MEG J / C / 1 -

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, 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> HarrisCorner(char*
input, char* output, int width, int
height, int channels, float
threshold, float k, float sigma)

支持 P5 和 P6 格式的 PPM 文件,从返回值数组中的第一个元素开始,返回值以 3 个元素为一组,分别是角点的 X 坐标、Y 坐标和分数,若返回值数组名为 A,则{A[0],A[1],A[2]}是第一个角点的数据,{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。

double* TemplateMatching1(char* inputFile, char* templateFile, char* output, char* output_txt, double threshold, int isWriteImageResult, unsigned char color, unsigned char red, unsigned char green, unsigned char blue)

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 是匹配结果图像名,output_txt 是保存的匹配相关数据的文本文件,threshold=0.5,isWriteImageResult=1,color是当图像是灰度图时的匹配框颜色,red、green和 blue 是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。

vector<Point>
TemplateMatching2(char*
inputFile, char*
templateFile, char* output, char*
output_txt, double threshold, int
isWriteImageResult, unsigned char
color, unsigned char red, unsigned
char green, unsigned char blue)

模板匹配,返回位置数组。参考: output 是 匹 配 结 果 图 像 名 ,output_txt 是保存的匹配相关数据的 文 本 文 件 , threshold=0.5 , isWriteImageResult=1, color 是当图像是灰度图时的匹配框颜色,red、green 和 blue 是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。

需要引入以下结构体:

double* TemplateMatching1(char* inputFile, char* templateFile, char* output_txt, double threshold, unsigned char color, unsigned char red, unsigned char green, unsigned char blue)

vector<Point>
TemplateMatching2(char*

typedef struct Point
{
 int x;
 int y;
}Point;

描板匹配 设同估数组, 匹配标为

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 是保存的匹配相关数据的文本文件,threshold=0.5 ;WriteImageResult=1,color是当图像是灰度图时的匹配框颜色,red、green和 blue 是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。

TemplateMatching2(char*
inputFile, char*
templateFile, char*
output_txt, double
threshold, unsigned char
color, unsigned char red, unsigned
char green, unsigned char blue)

模板匹配,返回位置数组。参考: output 是 匹 配 结 果 图 像 名,output_txt 是保存的匹配相关数据的文 本 文 件 , threshold=0.5 , isWriteImageResult=1, color 是当图像是灰度图时的匹配框颜色,red、green 和 blue 是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。

需要引入以下结构体:
typedef struct Point
{
 int x;
 int y;

}Point;

double* TemplateMatching4(char* inputFile, char* templateFile, char* output, char* output_txt, double threshold, int isWriteImageResult, unsigned char color, unsigned char red, unsigned char green, unsigned char blue)

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 是匹配结果图像名,output_txt 是保存的匹配相关数据的文本文件,threshold=0.5,isWriteImageResult=1,color 是当图像是灰度图时的匹配框颜色,red、green和blue是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持 PPM文件。

vector<Point>
TemplateMatching7(char*
inputFile, char*
templateFile, char* output, char*

模板匹配,返回位置数组。参考: output_txt 是保存的匹配相关数据的 文本文件, threshold=0.5, isWriteImageResult=1,color是当图 output_txt, double threshold, int isWriteImageResult, unsigned char color, unsigned char red, unsigned char green, unsigned char blue)

像是灰度图时的匹配框颜色, red、green 和 blue 是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。

需要引入以下结构体:
typedef struct Point
{
 int x;
 int y;

}Point;

double* TemplateMatching5(char*
inputFile, char*
templateFile, char*
output_txt, double
threshold, unsigned char
color, unsigned char red, unsigned
char green, unsigned char blue)

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的匹配框颜色,red、green和blue是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持PPM文件。

vector<Point>
TemplateMatching7(char*
inputFile, char*
templateFile, char*
output_txt, double
threshold, unsigned char
color, unsigned char red, unsigned
char green, unsigned char blue)

模板匹配,返回位置数组。参考: output_txt 是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的匹配框颜色, red、green和 blue 是当图像是彩色图时的匹配框颜色的红绿蓝通道值。支持 PPM 文件。

需要引入以下结构体:
typedef struct Point
{
 int x;
 int y;
} Point;

double* TemplateMatching(char*
inputFile, char*
templateFile, char*
output, unsigned char red, unsigned
char green, unsigned char
blue, double c, double
threshold, int a, int b1, int b2, int
c1, int c2)

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, a=20, b1=-30, b2=30, c1=5, c2=20。支持 PPM 文件。

double* TemplateMatching2(char*
inputFile, char*

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, a=20, b1=-30, b2=30, templateFile, unsigned char c1=5, c2=20。支持 PPM 文件。 red, unsigned char green, unsigned blue, double c, double threshold, int a, int b1, int b2, int c1, int c2) vector <double> 模板匹配,返回以匹配框的中心点坐 TemplateMatching3(char* 标、旋转角度和缩放比例组成的按顺 序存储的数组。参考: c=0.5, inputFile, char* templateFile, unsigned threshold=0.9, a=20, b1=-30, b2=30, char c1=5, c2=20。支持 PPM 文件。 red, unsigned char green, unsigned blue, double c, double char threshold, int a, int b1, int b2, int c1, int c2) 特征检测,返回特征点数量。支持 PPM unsigned int FeatureDetectionNumber(char* 文件。 input, char* output, unsigned char red, unsigned char green, unsigned char blue) 返回特征点检测结果。 std::list<sift::SiftKeypoint> 需引入以下结构体: FeatureDetection(char* input, char* output, unsigned char SIFT 关键点: 128 维 red, unsigned char green, unsigned namespace sift { char blue) #define DEGREE OF DESCRIPTORS (128)struct SiftKeypoint { int octave; // octave 数量 // laver 数量 int layer; float rlayer; // layer 实际数 量 // 归一化的 row float r; 坐标 float c; // 归一化的 co1 坐标 float scale; // 归一化的scale // row 坐 float ri; 标(layer) float ci; // column 坐标(layer) float layer scale; // scale (layer) float ori; // 方向(degrees).

float mag; // 模值

float

descriptors[DEGREE_OF_DESCRIPTOR
S];

};

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。

unsigned int
FeatureMatching1(char*
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(char*

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(char*
input, char* output, unsigned char
red, unsigned char green, unsigned

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。

char blue, bool	
bExtractDescriptor)	
std::list <ezsift::siftkeypoint></ezsift::siftkeypoint>	返回特征点列表。input 是输入图像,
FeatureExtraction2(char*	output 是生成的特征点图像,
input, char* output, unsigned char	bExtractDescriptor=true。支持 PGM
red, unsigned char green, unsigned	文件。
char blue, bool	需引入以下命名空间:
bExtractDescriptor)	namespace ezsift {
	#define DEGREE (128) //SIFT 关
	键点: 128 维
	struct SiftKeypoint {
	int octave; // octave 数量
	int layer; // layer数量
	float rlayer; // layer 实际数
	量
	float r; // 归一化的 row
	坐标
	float c; // 归一化的 col
	坐标
	float scale; // 归一化的 scale
	float ri; // row坐
	标(layer)
	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
	} ;
vootor/int\ Dofootloo-ti(-1	了 本比纳购位置 毛同母蚁烛奶妈位置
vector <int> DefectLocation(char*</int>	查找缺陷位置,返回材料缺陷的位置,
TemplateFile, char* SampleFile, int	以每 4 个元素为一组。TemplateFile
floor, int size, int a, int b, int	是模板图像,SampleFile 是样本图像,
c, int d, int e, int f, int g, int	g 是缺陷的有效界限的 X 轴长度,h 是 每路的有效界限的 X 轴长度,参考。
h, int FULL, int EMPTY, bool report)	缺陷的有效界限的 Y 轴长度,参考: floor=80 , size=10 , a=64, b=64 ,
	c=16, d=16, e=2, f=4, g=65, h=65,
	FULL=0,EMPTY=255, report=true。 支持 P5 格式的 PGM 文件。
vector <int> DefectSize(char*</int>	
TemplateFile, char* SampleFile, int	组。Template 是模板图像,Sample 是
c1	组。Telliplate 处探放图像,Salliple 处

floor, int size, int a, int b, int 样本图像, g是缺陷的有效界限的 X 轴

c, int d, int e, int f, int g, int	长度, h 是缺陷的有效界限的 Y 轴长
_	度, 参考: floor=80, size=10,
h, int FULL, int EMPTY, bool report)	
	a=64, b=64, c=16, d=16, e=2, f=4,
	g=65 , h=65 , FULL=0, EMPTY=255 ,
	report=true.
	支持 P5 格式的 PGM 文件。
vector <int> GoodBadQuantity(char*</int>	样品好坏的数量,返回结果中,第一个
TemplateFile, char* SampleFile, int	元素是合格的圆圈数量,第二个元素
floor, int size, int a, int b, int	是缺陷的圆圈数量。Template 是模板
c, int d, int e, int f, int g, int	图像,Sample 是样本图像,g 是缺陷的
h, int FULL, int EMPTY, bool report)	有效界限的 X 轴长度, h 是缺陷的有效
	界限的 Y 轴长度,参考: floor=80,
	size=10, a=64, b=64, c=16, d=16, e=2,
	f=4, g=65, h=65, FULL=0, EMPTY=255,
	report=true。
	支持 P5 格式的 PGM 文件。
void RAWSobelEdge(char*	Sobel 算子, input 是输入文件名,
input, char* output, int ROWS, int	output 是输出文件名。ROWS 是图像的
COLS, int M, float	行, COLS 是图像的列, M 是滤波相关参
sobelX[3][3], float sobelY[3][3])	数,如 M=1。支持 RAW 图像。
	参考模板:
	float sobelX[3][3] = {{-1,0,1},
	{-
	[2,0,2],
	{_
	1, 0, 1}};
	1, 0, 1)) ,
	float sobelY[3][3] = {{-1,-
	2,-1},
	2, 1),
	{0, 0, 0},
	(0, 0, 0),
	$\{1, 2, 1\}\};$
void RAWPlaceHolder(char*	边缘检测, input 是输入文件名,
input, char* output, int ROWS, int	output 是输出文件名。ROWS 是图像的
COLS, int M, float mask[3][3])	for COLS 是图像的列, M 是滤波相关参
COLS, III M, IIOat Mask[3][3]/	11,COLS 定图像的列,M 定滤板相大多 数,如 M=1。支持 RAW 图像。
	参考模板:
	float mask[3][3] = $\{\{-1, -2, -1\},\$
. 1	{0,0,0}, {1,2,1}};
void	拉普拉斯锐化滤波器,input 是输入文
RAWLaplacialSharpeningFilter(cha	件名,output 是输出文件名。ROWS 是
r* input, char* output, int	图像的行大小,COLS 是图像的列大小,
ROWS, int COLS, int M, float w, float	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。
mask[3][3])
                               参考模板:
                               float mask[3][3] = \{\{0, 1, 0\},\
                                                  \{1, -4, 1\},\
                                                  \{0, 1, 0\}\};
void
                               拉普拉斯算子增强, input1 是输入的
                               RAW图像文件名,output1是输出的RAW
RawLaplacianEnhancement(char*
                               图像文件名, width 是输入图像的宽,
input1, char*
                   output1, int
width, int height)
                               height 是输入图像的高。支持 RAW 图
            hough param circle*
                               圆检测,返回找到的圆的位置和大小
struct
                               等相关信息。支持 RAW 文件。
CircleDetection1(char* input, int
width, int height)
                               需引入以下结构体:
                               struct point {
                                  int x;
                                  int y;
                               };
                               struct hough param circle {
                                  int a;
                                  int b;
                                  int radius;
                                  int resolution;
                                  int thresh:
                                  struct point *points;
                                  int points size;
                               圆检测, 返回以找到的圆的位置和大
vector<struct
                               小等相关信息组成的数组。支持RAW文
hough param circle*>
CircleDetection2(char* input, int
                               件。
                               需引入以下结构体:
width, int height)
                               struct point {
                                  int x;
                                  int y;
                               } :
                               struct hough param circle {
                                  int a;
                                  int b:
                                  int radius;
                                  int resolution:
                                  int thresh;
                                  struct point *points;
                                  int points_size;
                               };
                               圆样变形,支持 RAW 文件。
void
```

```
ImageWarpEllipticalGrid(string
input, string
                     output, int
height, int
                      width, int
NumberChannels)
                                直线检测,返回直线的 theta 和 rho,
      FindLinel(char*
                      input, int
                                支持 RAW 图像。
width, int height)
                                直线检测,返回直线检测结果,支持
                   hough param*
struct
FindLine2(char* input, int width,
                                RAW 图像。
                                需引入以下结构体:
int height)
                                struct point {
                                    int x;
                                    int y;
                                };
                                struct hough param {
                                    int theta;
                                    int rho;
                                    int nrho;
                                    int mag:
                                    int resolution;
                                    int thresh;
                                    struct point *points;
                                    int points_size;
                                直线检测,返回直线检测结果数组,支
vector<struct
                  hough param*>
FindLine3(char* input, int width,
                                持 RAW 图像。
                                需引入以下结构体:
int height)
                                struct point {
                                    int x;
                                    int y;
                                };
                                struct hough param {
                                    int theta;
                                    int rho;
                                    int nrho;
                                    int mag;
                                    int resolution;
                                    int thresh:
                                    struct point *points;
                                    int points size;
int* FindCircle1(char* input,int
                                圆检测, 返回圆心的坐标和圆的半径,
                                支持 RAW 图像。sigma=1.4, tmin=70,
width, int height, float sigma, int
tmin, int tmax)
                                tmax=150.
                                圆检测,返回圆检测结果,支持 RAW 图
struct
             hough param circle*
```

```
FindCircle2(char*
                                 像。sigma=1.4, tmin=70, tmax=150。
                            int
                   input,
                                 需要引入以下结构体:
width, int height, float sigma,
int tmin, int tmax)
                                 struct point {
                                    int x;
                                    int y;
                                 };
                                 struct hough param circle {
                                    int a;
                                    int b:
                                    int radius;
                                    int resolution:
                                    int thresh:
                                    struct point *points;
                                    int points size;
                                 };
                                 圆检测,返回圆检测结果数组,支持
vector<struct
                                 RAW 图像。sigma=1.4, tmin=70,
hough param circle*>
FindCircle3(char*
                                 tmax=150.
                   input,
                            int
                                 需要引入以下结构体:
width, int height, float sigma,
int tmin, int tmax)
                                 struct point {
                                    int x;
                                    int y;
                                 };
                                 struct hough_param_circle {
                                    int a:
                                    int b:
                                    int radius;
                                    int resolution;
                                    int thresh;
                                    struct point *points;
                                    int points size;
           CornerDetection(char*
                                 角点检测, threshold=10000, k=0.06,
void
input, char*
                    output, float
                                 sigma=1.0, width=640, height=480,
                                 channels=1。支持 PNM 图像。
threshold,
             float
                     k,
                          float
sigma, int width, int height, int
channels)
                                 角点检测,返回角点数据。
vector < Keypoint >
CornerDetection1(char*
                                 threshold=10000 ,
                                                      k=0.06
input, char*
                                 sigma=1.0, width=640, height=480,
                    output, float
                                 channels=1。支持PNM图像。
threshold,
                          float
             float
                     k,
                                 需引入以下结构体
sigma, int width, int height, int
                                 typedef struct {
channels)
                                    float x;
```

float y; float score; } Keypoint; vector < Corner:: Keypoint > 角点检测,返回角点数据。 CornerDetection(char* input, int threshold=2000, k=1, sigma=1.2。支 width, int height, int 持 PNM 图像。 需引入以下命名空间: channels, float threshold, float k, float sigma) namespace Corner { struct Keypoint { float x; float y; float score; }; void Structure (char* input, char* 特征归一化统计,参考: sigma=2。支 持多种图像格式。 output, float sigma) 角点检测,参考: sigma=2, method=0。 void Cornerness(char* input, char* 支持多种图像格式。 output, float sigma, int method) 角点检测,参考: sigma=2, void Corners(char* input, char* output, float sigma, float thresh, thresh=0.4, window=5, nms=3, corner method=0。支持多种图像格 window, int int nms, corner method) 式。 void FindMatch(char* input1, char* 特征匹配,参考: thresh3=5, k=10000, input2, char* output, float cutoff=50, thresh4=5, sigma=2, thresh3, int k, int cutoff, float thresh=0.4, window=5, nms=3, thresh4, float float corner method=0 , sigma1=2, sigma, thresh, int window, int nms, int thresh1=0.4, window1=5, nms1=3, corner method, float sigmal, float corner method1=0 , sigma2=2, thresh1, int window1, int nms1, thresh2=0.4, window2=5, nms2=3, corner_method2=0 , int corner method1, float sigma2, sigma5=2, float thresh2, int window2, int thresh5=0.4, window5=7, nms5=3, corner_method5=0 , nms2, int corner method2, float sigma6=2, float $corner_method6=0$, thresh6=0.3, sigma5, thresh5, int window5, int nms5, window6=7 nms6=3int inlier thresh6=5, iters6=1000, corner method5, float sigma6, int float cutoff6=50, acoeff6=0.5。支持多种 corner method6, thresh6, int window6, int nms6, 图像格式。 float inlier_thresh6, int iters6, int cutoff6, float acoeff6) 角点检测,返回检测结果。参考: vector (Descriptor) HarrisCorner(char* sigma1=2, thresh1=0.4, window1=5, input, char* output, float sigmal, nms1=3, corner method1=0。支持多种 float thresh1, int window1, int nms1, 图像格式。 int corner method1) 需引入以下结构体:

```
struct Point
                                   double x, y;
                                   Point(): x(0), y(0) {}
                                   Point(double x, double y) :
                                 x(x), y(y) {}
                                 }:
                                 struct Descriptor {
                                   Point p:
                                   vector<float> data;
                                   Descriptor() {}
                                   Descriptor(const Point& p) :
                                 p(p) {}
                                 };
vector < Match >
                                 描述匹配项,返回描述结果。参考:
MatchDescriptors(char*
                                 sigma1=2, thresh1=0.4, window1=5,
input1, char*
                    input2, char*
                                 nms1=3 , corner method1=0 ,
                                 sigma2=2, thresh2=0.4, window2=5,
output, float
               sigmal,
                           float
thresh1, int window1, int nms1,
                                 nms2=3, corner method2=0。支持多种
int corner method1, float sigma2,
                                 图像格式。
float thresh2, int window2, int
                                 需引入以下结构体:
nms2, int corner method2)
                                 struct Point {
                                   double x, y;
                                   Point(): x(0), y(0) {}
                                   Point (double x, double y) :
                                 x(x), y(y) {}
                                 };
                                 struct Descriptor {
                                   Point p;
                                   vector<float> data;
                                   Descriptor() {}
                                   Descriptor(const Point& p) :
                                 p(p) {}
                                 }:
                                 struct Match {
                                   const Descriptor* a=nullptr;
                                   const Descriptor* b=nullptr;
                                   float distance=0.f;
                                   Match() {}
                                   Match (const Descriptor* a, const
```

Descriptor* b, float dist=0.f) :
a(a), b(b), distance(dist) {}

bool operator<(const Match&
other) { return
distance<other.distance; }
}.</pre>

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

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

void PanoramaImage(char* input1, char* input2, char* output, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float threshl, int windowl, int corner method1, float nms1, sigma2, float thresh2, int window2, nms2, int int corner method2, float sigma5, float thresh5, int window5, nms5, int 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=3, corner method5=0 , sigma6=2, corner method6=0, thresh6=0.3, window6=7 nms6=3inlier thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5。支持多种 图像格式。

 角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4,

thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int corner method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float 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, cutoff6, float acoeff6, float sigma7, int corner_method7, float thresh7, int window7, int nms7, float inlier thresh7, int iters7, int cutoff7, float acoeff7)

window=5, nms=3, corner method=0, sigma1=2, thresh1=0.4, window1=5, corner method1=0 sigma2=2, thresh2=0.4, window2=5, nms2=3, corner_method2=0 sigma5=2, thresh5=0.4, window5=7, nms5=3, corner method5=0 sigma6=2, corner method6=0 thresh6=0.3, window6=7, nms6=3, inlier thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5, sigma7=2, corner method7=0, thresh7=0.3, window7=7 nms7=3inlier thresh7=5, iters7=1000, cutoff7=50, acoeff7=0.5。支持多种 图像格式。

void Spherical (char* input1, char* input2, char* output, float f1, float f2, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int corner method1, float sigma2, float thresh2, int window2, int int corner method2, float nms2, sigma5, float thresh5, int window5, int nms5. int corner method5, float sigma6, int corner_method6, float thresh6, nms6, int window6, int float inlier_thresh6, int iters6, int cutoff6, float acoeff6, float sigma7, int corner method7, float thresh7, int window7, int nms7, float inlier_thresh7, int iters7, int cutoff7, float acoeff7)

角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3 , corner method1=0 sigma2=2, thresh2=0.4, window2=5, corner_method2=0 nms2=3 , sigma5=2, thresh5=0.4, window5=7, nms5=3, corner_method5=0 sigma6=2, corner method6=0, thresh6=0.3, window6=7, nms6=3, inlier thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5, sigma7=2, corner_method7=0, thresh7=0.3, window7=7 nms7=3inlier_thresh7=5, iters7=1000, cutoff7=50, acoeff7=0.5。支持多种 图像格式。

std::vector <MATCHED_POINT *>
FeatureMatching(char*
input1, char* input2, char* output)

特征匹配,支持 PGM 图像。 需引入以下结构体: typedef struct matchedpoint

	(
	. 1
	unsigned int uiPosXFirst;
	unsigned int uiPosYFirst;
	unsigned int uiPosXSecond;
	unsigned int uiPosYSecond;
	bool bIsSample;
	}MATCHED_POINT;
int FeatureMatching(char*	特征匹配,返回匹配的点对,支持 BMP
input1, char* input2, char*	图像。
output, char* outputCouple)	
std::vector <centerspoint></centerspoint>	圆检测,返回检测结果。size1=5,
FindCircles(char* input, char*	size2=5,size3=7。支持 BMP 图像。
output, int size1, int size2, int	需引入以下结构体:
size3)	struct Point {
	Point(int $x = 0$, int $y = 0$)
	$\{ this \rightarrow x = x; this \rightarrow y = y; \}$
	int x;
	int y;
	};
	struct CentersPoint {
	CentersPoint(Point point, int
	radius) { this->point = point;
	this->radius = radius; count =
	1; }
	Point point;
	int count;
	int radius;
	}.
void Canny(char* input, char*	Canny 算子,至少支持 JPG 图像,input
output, int lowThreshold, int	是输入文件名, output 是输出文件名,
highThreshold)	参考: lowThreshold=50,
inighthi eshotu)	highThreshold=150.
void KMeans1(char* input,char*	K-Means 聚类,input 是输入文件名,
` '	<u> </u>
output, int c, int k)	output 是输出文件名。输入图像最好
	宽高相同,c的最大值是图像的宽和高 中校小的那么参数。如第5500 京为
	中较小的那个参数,如宽=500,高为
	600,则 c 最大可取 500; k 是聚类的
1 1714	种类数目。支持 BMP 图像。
void KMeans(string input, unsigned	K-Means 聚类,input 是输入文件名,
int Clusters, char* output)	Clusters 是聚类的种类数目,output
11 1000 1/1	是输出文件名。支持 BMP 图像。
void LSBRead(char* input, char*	LSB 隐写文件的读出, input 是输入文
output, int width, int	件名,output 是输出文件名。参考:
height, unsigned char	color1=255,color2=0。支持 24 位 BMP

color1, unsigned char color2)	图像。
void LSBWrite(char* input1, char*	LSB 隐写,input1 是用于容纳隐藏图
input2, char* output, int width, int	像的图像,input2 是要隐藏的黑白图
height, unsigned char	像,参考: threshold1、threshold2 和
thresholdl, unsigned char	threshold3 都 等 于 128 ,
threshold2, unsigned char	color1=(unsigned char)
threshold3, unsigned char	0b00000001, color2=(unsigned char)
color1, unsigned char color2)	0b11111110。支持 24 位 BMP 图像。
void STLSection(char* input, char*	STL 切片,input 是输入的 STL 文件,
output, int sliceAmount, int	output 是输出的切片文件前缀名,
resolution, int c)	sliceAmount 是切片量,如:
	sliceAmount=50, resolution 是分辨
	率,如:resolution=260,c是执行的
	相关参数,如: c=5。
void SURF(char* input1, char*	SURF 算子,input1 和 input2 是输入
input2, char* output)	文件名,output 是输出文件名。支持
	BMP 图像。
void SobelBinary(char*	二进制法 Sobel 算子, input1 和
input, char* output, char	input2 是输入文件名,output 是输出
filterH[9], char filterV[9])	文件名。支持 BMP 图像。
	参考:
	$char filterH[9] = \{-1, 0, 1,$
	-2, 0, 2,
	-1, 0, 1;
	char filterV[9] = {1, 2, 1,
	0, 0, 0,
	-1, -2, -
	1};
void SobelOperator(char*	Sobel 算子,耗时较长,input 是输入
input, char* output)	文件名, output 是输出文件名。支持
	24 位 BMP 图像。
SobelImage** SobelOperator(char*	返回处理后的各像素点的坐标和对应
input)	的像素值, 若是边缘点则对应白色, 否
	则对应黑色。支持 BMP 图像。
	需引入以下结构体:
	typedef struct {
	int x;
	int y;
	unsigned char red;
	unsigned char freen;
	unsigned char blue;
	SobelImage;
void Sobel2(char* input, char*	Sobel 算子,input 是输入文件名,
· · · · · · · · · · · · · · · · · · ·	
output, char filterH[9], char	output 是输出文件名。支持 BMP 图像。

filterV[9])	参考:
111te1v[J])	char filterH[9] = {-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1};
	char filterV[9] = {1, 2, 1,
	0, 0, 0,
	-1, -2, -
	1};
void EdgeDetection(char*	边缘检测,input 是输入文件名,
	output 是输出文件名。支持 4 位 BMP
input, char* output)	图像。
void EdgeDetection1(char*	边缘检测, input 是输入文件名,
input, char* output, short	output 是输出文件名。支持 8 位 BMP
sharpen[3][3])	图像。
	参考模板:
	short sharpen[3][3] = $\{\{1, 1, 1\},$
	{1, -8,
	1},
	{1, 1,
	1}};
void EdgeDetection2(char*	边缘检测, input 是输入文件名,
input, char* output, int a)	output 是输出文件名。a 是用于设置
	图像像素的相关参数,如 a=3。支持 24
	位 BMP 图像。
void EdgeDetection3(char*	边缘检测, input 是输入文件名,
input, char* output, int a)	output 是输出文件名。a 是用于设置
	图像像素的相关参数,如 a=3。支持 24
	位 BMP 图像。
void EdgeDetection4(char*	边缘检测, input 是输入文件名,
input, char* output, int a)	output 是输出文件名。a 是用于设置
	图像像素的相关参数,如 a=3。支持 24
	位 BMP 图像。
<pre>void Roberts(char* input, char* output)</pre>	Roberts 边缘检测。支持 BMP 图像。
void Prewitt(char* input, char*	Prewitt 边缘检测。支持 BMP 图像。
output)	
void Sobel(char* input, char*	Sobel 边缘检测。支持 BMP 图像。
output)	
void Laplace(char* input, char*	Laplace 边缘检测。支持 BMP 图像。
output)	1
void QRCodeGeneration(char	二维码生成,filename 是生成的二维
filename, char inputString)	一码图像文件名, inputString 是二维码
	包含的信息。支持 BMP 图像。
void QRCodeGenerate(string	二维码生成,output 是生成的二维码
, ora whoodedenerate (String	→ >下は1丁>M2 on obu o 1/一>下出

inputString, char* output)	图像文件名, inputString 是二维码包
000 10 1 / 1	含的信息。支持 BMP 图像。
void QRCodeEncode(char*	二维码编码,输入文本文件,输出二维
input, char* output)	码的 PBM 图像。
void QRCodeDecode(char*	二维码解码,输入为二维码的 PBM 图
input, char* output)	像,输出为文本文件。
detectie* TemplateMatching1(char*	模板匹配,返回各个目标的位置等数
input, char* Template, char*	据,MatchScore=0.6,k=0。支持24位
output, float MatchScore, pixel	BMP 图像。
C, unsigned int k)	需引入以下结构体:
	typedef struct { unsigned char R;
	unsigned char
	G;
	unsigned char
	B;}pixel;
	typedef struct { unsigned int x;
	unsigned int
	у;
	float prag;
	pixel culoare;
	} detectie;
detectie**	模板匹配,返回各个目标的位置等数
TemplateMatching2(char*	据, MatchScore=0.6, k=0。支持24位
input, char* Template, char*	BMP图像。
output, float MatchScore, pixel	需引入以下结构体:
C, unsigned int k)	typedef struct { unsigned char R;
	unsigned char
	G;
	unsigned char
	B;}pixel;
	typedef struct { unsigned int x;
	unsigned int
	у;
	float prag;
	pixel culoare;
	} detectie;
struct Template*	模板匹配,a=0.5,b=0,a1=0.5,
TemplateMatching(char*	b1=0.5, c=0.2, ps 是相似度, 如
input, char* Template, char*	ps=0.5。支持 24 位 BMP 图像。
output, unsigned int b, double	需引入以下结构体:
ps, double a, double a1, double	struct Template
b1, double c)	1.11 . 1.41
	double corelatia;
	unsigned int nrFereastra;

```
unsigned int cifra;
                                  unsigned int ly1;
                                  unsigned int rly;
                                  unsigned int 1x1;
                                  unsigned int rlx;
vector < Search Result >
                               模板匹配,返回以目标位置的左上角
                               坐标和相似值组成的数组。
TemplateMatching(char*
input, char* Template)
                               支持 PNG 图像。
                               需引入以下结构体:
                               struct SearchResult {
                                   int x, y;
                                   double value;
                               };
void
        TemplateMatching1(char*
                               模板匹配, ps 是相似度, ps=0.8,
input, char*
                Template, char*
                               colors=(image\_colors) \{255, 0, 0\}.
output, float
               ps, image colors
                               支持24位BMP图像。
colors)
                               需引入以下结构体:
                               typedef struct {
                                  unsigned char R, G, B;
                               } image_colors;
        TemplateMatching2(char*
                               模板匹配, input 是输入的 8 位 BMP 图
void
                               像, Template 是 24 位模板图像, ps 是
input, char*
                Template, char*
output, float
                               相
                                    似
                                         度
                                                  ps = 0.8
               ps, image colors
                               colors = (image\_colors) \{255, 0, 0\} .
colors)
int*
         TemplateMatching(char*
                               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
input1, char*
                  input2, char*
                               匹配框左下角的纵坐标, 第三个元素
output, unsigned char red, unsigned
                               是匹配框的左上角的横坐标, 第四个
        green, unsigned
                          char
char
                               元素是匹配框的右上角的横坐标。
blue, double MatchScore)
                               input1 是搜索图像, input2 是模板图
                               像, output 是匹配结果图像,
                               MatchScore=0.9.
                               模板匹配,返回目标的位置等数据。
                     vFerestre
struct
TemplateMatchingl(char*
                               input1 是搜索图像, input2 是模板图
                               像, output 是匹配结果图像,
input1, char*
                  input2, char*
output, unsigned char red, unsigned
                               MatchScore=0.9.
                               需引入以下结构体:
        green, unsigned
char
                         char
blue, double MatchScore)
                               //窗口结构
                               typedef struct fereastra {
                                  char seAfiseaza;
                                   int iStart, iFinal, jStart,
                               jFinal;
```

```
double cor;
                             };
                             //窗口向量结构
                              typedef struct vFerestre {
                                 int nr:
                                 struct fereastra *v;
                             模板匹配,返回值中第一个元素是匹
int*
         TemplateMatching(char*
                             配框左上角的纵坐标, 第二个元素是
input1, char* input2, unsigned char
red, unsigned char green, unsigned
                             匹配框左下角的纵坐标,第三个元素
char blue, double MatchScore)
                              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.
void
         TemplateMatching(char*
                             模板匹配, suprapunereMaxima 表示最
input, char*
            templatename, char*
                                     叠
                                         率
                             大
output, unsigned
                             MaximumMatchingQuantity=10
                         int
MaximumMatchingQuantity, double
                             MatchScore=0.8
MatchScore, float
                             suprapunereMaxima=0.2.
suprapunereMaxima, unsigned
                        char
red, unsigned char green, unsigned
char blue)
vector<int>
                              模板匹配,支持 24 位 BMP 图像,返回
TemplateMatching(char*
                              以匹配框的左上角坐标(x, y)组成的
input, char* Template)
                              按顺序存储的数组。
                             模板匹配,返回值是匹配框的左上角
int*
         TemplateMatching(char*
input1, char* input2, char* output)
                             坐标 (x, y)。
                              模板匹配,返回值是匹配框的左上角
vector(coord)
TemplateMatchingl(char*
                             坐标组成的向量。
input1, char* input2, char* output)
                              需引入一下结构体:
                             struct coord {
                                 int x = 0;
                                 int y = 0;
                             };
        TemplateMatching4(char*
                             模板匹配,返回值是匹配框的左上角
int*
input1, char* input2)
                              坐标(x,y)。
vector<coord>
                              模板匹配,返回值是匹配框的左上角
TemplateMatching3(char*
                              坐标组成的向量。
                              需引入一下结构体:
input1, char* input2)
                             struct coord {
                                 int x = 0;
                                 int y = 0;
```

vector<int> 模板匹配,返回值是匹配框的左上角 坐标(x, y)组成的向量,按顺序以两个 TemplateMatching(char* 为一组。min 是与匹配分数相关的参 input1, char* input2, float min) 数,参考: min=65026。 vector<TargetData> 模板匹配,支持 JPG 图像,返回值的第 1 和第 2 个元素是匹配框左上角顶点 TemplateMatch(char* inputfile, char* 的横坐标和纵坐标,第3个元素是目 标与模板相对的旋转角度,第4个元 templatefile, char* output, int 素是缩放比例。参考: size=1, size, int best loss, double a, double b, double c, double d, int best loss=1000000000, a=0.5, e1, int e2) b=2.1, c=0.5, d=45, e1=20, e2=20. 需引入一下结构体: typedef struct TargetData{ int best x; int best y; double best r; double best s; } TargetData; 模板匹配,支持 JPG 图像,返回值的第 vector<TargetData> 1 和第 2 个元素是匹配框左上角顶点 TemplateMatch(char* inputfile, char* templatefile, int 的横坐标和纵坐标,第3个元素是目 size, int best loss, double 标与模板相对的旋转角度,第4个元 素是缩放比例。参考: size=1, a, double b, double c, double d, int e1, int e2) best loss=1000000000, a=0.5, b=2.1, c=0.5, d=45, e1=20, e2=20. 需引入一下结构体: typedef struct TargetData{ int best x; int best y; double best r; double best s; } TargetData; 模板匹配,返回匹配到的目标数量。 ObjectFind(char* int inputfile, char* Templatefile) 图像匹配,返回值中的前10个元素是 float* ImageMatching(char* 按顺序对应的目标与模板的差异分 TargetImage, char* TemplateO, char* 数,最后一个元素是匹配到的模板的 Templatel, char* Template2, char* Template3, char* Template4, char* 序号。 Template5, char* Template6, char* Template7, char* Template8, char* Template9) 图像匹配,返回值中的前2个元素是 float* ImageMatching(char* 按顺序对应的目标与模板的差异分 Target Image, char* TemplateO, char* Template1) 数,最后一个元素是匹配到的模板的

void ImageFeatures(char*	序号。支持 BMP 图像。 图像特征。
input, char* kernel, char* output)	kernel 文件内容样例:
	3
	$\begin{bmatrix} 1 \\ 0 & -1 & 0 \end{bmatrix}$
	-1 5 -1
	$\begin{bmatrix} -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$
	0 -1 0 其中,3 表示尺寸为 3*3,1 表示内核
	共中,
void FileWrite(char* BMP,char*	图像隐写之文件写入,将文本文件写
· · · · · · · · · · · · · · · · · · ·	図像版与之文件与八, 将文本文件与 入图像。支持 32 位 BMP 图像。BMP 是
TXT)	
	要写入的图像文件名,TXT 是要写入图
woid FiloWriteOut (shows DMD -1	像的文本文件名。 图像图写文文件写山 收立太文件
void FileWriteOut(char* BMP, char* TXT)	图像隐写之文件写出,将文本文件从图像中取出来。支持 32 位 BMP 图像。
111)	BMP 是要写出的图像文件名,TXT 是写
	出图像后信息保存的文本文件名。
void LBP(char* input, char*	LBP 图像特征提取。支持 PNG 图像。
output)	LDI 图像行业泛水。文诗 ING 图像。
void Watershed2(char* input, char*	图像分割之分水岭算法。
inputMarqueurs, char* output, int	inputMarqueurs 是输入图像的标记图
r, unsigned char R, unsigned char	像。R=230,G=0,B=0,r=1。支持PNG
G, unsigned char B)	像。 k-230, b-0, b-0, i-i。 文持 i ko 图像。
void EcrireImage1(char*	图像分割。rayon=5。支持 PNG 图像。
input, char* output, uint32_t	Eligible 1 try on 50 Ziji Tho Eligib
rayon)	
void EcrireImage2(char*	图像分割。rayon=5。支持 PNG 图像。
input, char* inputMarqueurs, char*	Elevation of Military
output, uint32_t rayon)	
void EcrireLPECouleur1(char*	图像分割。rayon=5。支持 PNG 图像。
input, char* inputMarqueurs, char*	
output, uint32 t rayon)	
void Watershedl (char* input, char*	图像分割之分水岭算法。
inputMarqueurs, char*	inputMarqueurs 是输入图像的标记图
output, uint32 t rayon)	像。rayon=5。支持 PNG 图像。
void EcrireImage3(char*	图像分割。rayon=1。支持 PNG 图像。
input, char* inputMarqueurs, char*	244
output, uint16_t rayon)	
void	图像分割。rayon=1。支持 PNG 图像。
EcrireImageCouleursAleatoires(ch	
ar* input, char*	
inputMarqueurs, char*	
output, uint8_t r, uint8_t	

g,uint8_t b,uint16_t rayon)	
void Watershed(char* input, char*	图像分割。inputMarqueurs 是输入图
inputMarqueurs, char*	像的标记图像。a一般为255,rayon=1。
output, uint8_t r, uint8_t	支持 PNG 图像。
g, uint8_t b, uint8_t a, uint16_t	
rayon)	
void FloodFill(char* input, char*	图像分割之漫水填充法。x=0,y=0,
output, int x, int y, unsigned char	novaCor=127。支持 PGM 文件。
novaCor)	
void StyleTransfer(char*	图像风格迁移,iterations=10。至少
input1, char* input2, char*	支持 JPG 和 PNG 图像。
output, int iterations)	
void ImageFusion(char*	图像融合,支持 PNG 图像。
input1, char* input2, char*	
inputUniqe1,char*	
inputUniqe2, char* output)	
void Uniqe(char* input,char*	图像关键点标识, R=255, G=0, B=0。
inputUniqe, char* output, double	
R, double G, double B)	
void AVGFused(char* input1, char*	图像融合,支持 PGM 图像。
input2, char* output)	
void MAXFused(char* input1, char*	图像融合,支持 PGM 图像。
input2, char* output)	
void MINFused(char* input1, char*	图像融合,支持 PGM 图像。
input2, char* output)	
void FreqFused(char* input1, char*	图像融合,支持 PGM 图像。
input2, char* output)	
void	读取点坐标并输出点之间绘制的点或
ConvertCoordinatesToGraphics(cha	线段的图像。IMAGE_SIZE=800,
r* input, char* output, double	PIXEL PADDING=25, drawlines=0 或
IMAGE_SIZE, double PIXEL_PADDING,	drawlines=1.
bool drawlines)	输入文件格式
	如果我们将"N"表示为点数,则假设
	采用以下点坐标文件格式:
	N N N N N N N N N N N N N N N N N N 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
	····
maid Home Day 11 1/1	N x 坐标 y 坐标
void HumanDetection1(char*	人类检测。参考: MINH=0.0,
input, char* output, double MINH,	MAXH=50.0, MINS=0.23, MAXS=0.68.
double MAXH, double MINS, double	支持 24 位 BMP 图像。

MAXS)	
void HumanDetection2(char* input,	人类检测。参考: MINH=0.0,
char* output, double MINH, double	MAXH=50.0, MINS=0.23, MAXS=0.68。
MAXH, double MINS, double MAXS)	支持 24 位 BMP 图像。
void HumanDetection3(char* input,	人类检测。参考: MINH=0.0,
char* output, double MINH, double	MAXH=50.0, MINS=0.23, MAXS=0.68。
MAXH, double MINS, double MAXS)	支持 24 位 BMP 图像。
int ImageFeatureNumber(char*	计算图像的特征点数量。支持24位BMP
input)	图像。
int ContentSimilarity(char*	返回两个图像的内容相似度。
input1, char* input2, int	KDTREE_BBF_MAX_NN_CHKS 是在 BBF 搜
KDTREE_BBF_MAX_NN_CHKS, double	索期间要检查的关键点 NN 候选的最大
NN_SQ_DIST_RATIO_THR)	数量,NN_SQ_DIST_RATIO_THR 是 NN 和
	第二个 NN 之间距离平方比的阈值,参
	考: KDTREE_BBF_MAX_NN_CHKS=100,
	NN_SQ_DIST_RATIO_THR=0.49。支持24
	位 BMP 图像。
Feature* ImageFeature(char*	返回图像的特征数据。支持 24 位 BMP
input)	图像。
	需引入一下结构体:
	typedef struct Point2D64f
	{
	double x;
	double y;
	}Point2D64f;
	typedef struct feature
	{
	double x;
	/x 坐标/
	double y;
	/y 坐标/
	double a;
	/0xford 型仿射区域参数/
	double b;
	/0xford 型仿射区域参数/
	double c;
	/0xford 型仿射区域参数/
	double scl;
	/Lowe 风格特征的比例/
	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;
double CharacterRecognition(char*	字符匹配,支持 BMP 图像,返回值是目
TargetImage, char*	标图像匹配到的模板文件的序号,如
TemplateFileGroup[])	返回值是2则说明图像与序号为2(序
	号从零开始)的模板匹配。
	参 考:
	<pre>TemplateFileGroup[]={ "0.txt",</pre>
	"1. txt", "2. txt", "3. txt",
	"4. txt", "5. txt", "6. txt",
	"7. txt", "8. txt", "9. txt" };
double	字符匹配,支持 BMP 图像,返回值是目
CharacterRecognition1(char*	标图像匹配到的模板文件的序号,如
TargetImage, char*	返回值是2则说明图像与序号为2(序
TemplateFileGroup[])	号从零开始)的模板匹配。
	参 考:
	TemplateFileGroup[]={ "0.txt",
	"1. txt", "2. txt", "3. txt",
	"4. txt", "5. txt", "6. txt",
	"7. txt", "8. txt", "9. txt" };