#### 使用说明书

# 目录

PPM、PGM 和 PBM 图像处理	
<b>局</b> 级算于	

### PPM、PGM 和 PBM 图像处理

void OTSUBinarization(char*	OTSU 二值化。input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P5 格式的
	PGM 图像。
void PPMtoBMP(char* input, char*	PPM 图像转为 BMP 图像。bpp 是 BMP 图
output, int bpp)	像的色深。
<pre>void BMPtoPPM(char* input, char*</pre>	BMP 图像转 PPM 图像。
output)	
void PPMtoBMP1(char* input, char*	PPM 图像转为 BMP 图像。bpp 是 BMP 图
output, int bpp)	像的色深。
void BMPtoPPM1(char* input, char*	BMP 图像转 PPM 图像。
output)	
void BMPtoPGM(char* input, char*	BMP 转 PGM。
output)	
void BMPtoPPM2(char* input,	BMP 转 PPM。
char* output)	
void PPMtoPGM(char* input, char*	PPM 转 PGM。
output)	
void BlurPPM(char* input, char*	PPM 图像滤波。
output)	
void BlurPGM(char* input, char*	PGM 图像滤波。
output)	
void	OTSU二值化划分。input 是输入文件名,
SegmentsOTSUBinarization(char*	output 是输出文件名。支持 P5 格式的
input, char* output)	PGM 图像。
void P3PPMBlur(char* input, char*	PPM 图像模糊, input 是输入文件名,
output)	output 是输出文件名。支持 P3 格式的
	PPM 图像。
unsigned char** ReadPBM(char*	读取 PBM 图像并返回图像数据。input
input)	是要读取的 PBM 图像文件名。支持 P4
	格式的 PBM 图像。

: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)
	/ <b>-</b>
	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

:	子的 DDM 因俗 - 目\
input, char* output, int a)	式的 PPM 图像, a 是增亮系数, 如 a=80。
void GrayBrightening(char*	灰度图像增亮,输入和输出都是 PGM 图
input, char* output, int a)	像,a 是增亮系数,如 a=80。
void PPMFilter(char* input, char*	彩色图像滤波,输入和输出都是 P6 格
output)	式的 PPM 文件。
void PGMGrayFilter(char*	灰度图像滤波,输入和输出都是 PGM 图
input, char* output)	像。
void PPMtoBMP(char* input, char*	PPM 图像转 BMP 图像, input 是输入文
output)	件名,output 是输出文件名。支持 P6
	格式的 PPM 图像。
void PGMRotated(char*	图像旋转, channels 是输入图像的通
input, char* output, int width, int	道, theta 是旋转弧度, 支持 PGM 图像。
height, int channels, double	参考: theta=45.0*3.1415926/180。
theta)	
void XCorner(char* input, char*	X 方向角度, channels 是输入图像的通
output, int width, int height, int	道。支持 PGM 图像。
channels, double theta)	
void YCorner(char* input, char*	Y 方向角度, channels 是输入图像的通
output, int width, int height, int	道。支持 PGM 图像。
channels, double theta)	7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
void Smooth(char* input, char*	图像模糊, channels 是输入图像的通
output, int width, int height, int	道, sigma_x 是 X 方向的模糊系数,
channels, float sigma_x, float	sigma_y 是 Y 方向的模糊系数。
sigma_y, double theta)	
void PGMOtsuThreshold(string	大津阈值法, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 P5 格式的
	PGM 图像。
void	局部大津阈值, input 是输入文件名,
PGMLocalisedOtsuThreshold(string	output 是输出文件名。支持 P5 格式的
input, char* output)	PGM 图像。
void PGMSauvolaThreshold(string	索沃拉阈值,支持 P5 格式的 PGM 图像。
input, char* output, double	a、b和c的参考值如:
a, double b, double c)	a=0. 01, b=15, c=225.
void PGMThreshold(string	阈值法, input 是输入文件名, output
input, char* output, int thresh)	是输出文件名。支持 P5 格式的 PGM 图
	像。thresh 是阈值,如: thresh=5。
float Repair1(char* input, char*	图像修复, var 是噪声方差, threshold
output, float var, float	是阈值, nbLevels 是要处理的级别数,
threshold, int nbLevels, float a)	a=10。返回 ISNR。
float Repair2(char* input, char*	图像修复, var 是噪声方差, threshold
output, float var, float	是阈值,nbLevels 是要处理的级别数,
threshold, int nbLevels, float a)	a=10。返回 ISNR。
void LowPassFilterRepair1(char*	低通滤波图像修复, a=10, b=6,
input, char* output, int	nbLevels=3,size_filter 是低通过滤

size_filter, float var, int nb_iterations, int nbLevels, float a, int b)	器的大小, var 是噪声方差, nb_iterations 是 Landweber 的迭代 数。
<pre>void LowPassFilterRepair2(char* input, char* output, int size_filter, float var, int nb_iterations, int nbLevels, float</pre>	低通滤波图像修复,a=10,b=6,nbLevels=3,size_filter是低通过滤器的大小,var是噪声方差,nb_iterations是Landweber的迭代
a, int b)  float LowPassFilterRepair3(char* input, char* output, int size_filter, float var, int	数。 低通滤波图像修复,a=10,b=6, nbLevels=3,pas=1,size_filter是低 通过滤器的大小,var是噪声方差,
nb_iterations, int nbLevels, int pas, float a, int b) void Repair1(char* input, char* output, int M, float a)	nb_iterations 是 Landweber 的迭代数。返回 ISNR。 图像修复,a=0.0,M 是分解的层次数,如 M=3。
<pre>void Repair2(char* input, char* output, int M, float a) void MakeNoise1(char* input, char* output, int</pre>	图像修复,a=0.0,M是分解的层次数,如 M=3。 制造噪声,size_filter是低通滤波器的宽度。
<pre>size_filter) void</pre>	制造噪声,nb_iterations是Landweber的迭代数,pas=1。
<pre>nb_iterations, int pas) void MakeNoise3(char* output, int height, int width, float var)</pre>	制造噪声,height 是输出图像的高,width 是输出图像的宽,var 是噪声方差。
<pre>void</pre>	制造噪声,nb_iterations是Landweber 的迭代数,pas=1。
<pre>void ImageReconstruction(char* input, char* output, int maxDepth, int threshold, int tx, int ty)</pre>	图像重建,支持 PGM 文件。参考: maxDepth=80, threshold=50, tx=0, ty=0。

### RAW 图像处理

unsigned char** RAWRead(char*	读取 RAW 图像。
input, int height, int width)	
void RAWWrite(unsigned char**	保存 RAW 图像。
input, char* output, int height, int	
width)	
void MBVQ(char* input, char*	MBVQ 效果, input 是输入文件名,
output, int width, int height)	output 是输出文件名。width 和
	height 是输出图像的宽和高。
void RAWtoPPM_red(char*	RAW 转为 PPM 后提取红色通道,参考:

1 1 1 1 1	:1,1-4000 1 : 1 : 2070
<pre>input, char* output, int width, int height, DebayerAlgorithm algo)</pre>	width=4096 , height=3072 , algo=NEARESTNEIGHBOUR 或 LINEAR。
	支持 RAW12 格式。
	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR, LINEAR
	LINEAR
void RAWtoPPM green1(char*	RAW 转为 PPM 后提取绿色 1 通道,参
input, char* output, int width, int	考: width=4096, height=3072,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR或LINEAR。
nergitt, besatjerninger tellin arge)	支持 RAW12 格式。
	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	};
void RAWtoPPM_green2(char*	RAW 转为 PPM 后提取绿色 2 通道,参
input, char* output, int width, int	考: width=4096, height=3072,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR 或 LINEAR。
	支持 RAW12 格式。
	需引入以下枚举:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
void RAWtoPPM blue(char*	RAW 转为 PPM 后提取蓝色通道,参考:
input, char* output, int width, int	width=4096 , height=3072 ,
height, DebayerAlgorithm algo)	algo=NEARESTNEIGHBOUR或LINEAR。
nergitt, besageringer tilm argo)	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:
	enum DebayerAlgorithm {
	NEARESTNEIGHBOUR,
	LINEAR
	J;

input, char* output, int width, int height, int c, float v)  void RAWAvgFilter(char* input, char* output, int ROWS, int COLS, int M, float mask[3][3])  void RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RAWMedianFilter(char* input, char* output, int sequence[9])  void RawtoBmpl(char* input, char* output, unsigned long Width, \times RAW 图像 \times \ti	换,input 是输入的 RAW 图像
height, int c, float v	, output 是输出的 RAW 图像文
Void RAWAvgFilter(char* input, char* output, int ROWS, int COLS, int M, float mask[3][3])  Void RawImageInversion(char* input, char* output, int width, int height)  Void RawHistogramEqualization(char* input, char* output, int width, int height)  Void RAWHistogramEqualization(char* input, char* output, int width, int height)  Void RAWHistogramEqualization(char* input, char* output, int width, int height)  Void RAWHedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  Void RAWMedianFilter(char* input, char* output, int width, int height)  Void RAWMedianFilter(char* input, char* output, unsigned long Width, unsigned long Height)  RAW 图像 为 考模机 int sequence[9])	idth 是输入图像的宽, height
文持 RAW void RAWAvgFilter(char* input, char* output, int ROWS, int COLS, int M, float mask[3][3])  RawImageInversion(char* input, char* output, int width, int height)  void RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤影 output, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmpl(char* input, char* output, unsigned long Width, unsigned long Height)  Width 和 RAW 图像 output, unsigned long Width, unsigned long Width, unsigned long Width, unsi	
woid RAWAvgFilter(char* input, char* output, int ROWS, int COLS, int M, float mask[3][3])  woid RawImageInversion(char* input, char* output, int width, int height)  woid RawHistogramEqualization(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int width, int height)  woid RAWMedianFilter(char* input, char* output, int sequence [9])  woid RawtoBmpl(char* input, char* output, unsigned long Width, unsigned long Height)  width files way and wa	图像的高。默认 c=1, v=0.6。
input, char* output, int ROWS, int COLS, int M, float mask[3][3])  void RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmpl(char* input, char* output, unsigned long Width, unsigned long Width, width for the color output, unsigned long Width, width for the color output in the color output, unsigned long Width, width for the color output, unsigned long Width, width for the color output in the color output, unsigned long Width, width for the color output in the color o	
COLS, int M, float mask[3][3])  的方大人是滤波棒	波器, input 是输入文件名,
是滤波模	是输出文件名。ROWS 是图像
波器模构 参考模构 float {{0.1111} void RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawWoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)  void RAWWB Mint sequence[9]	小,COLS 是图像的列大小,M
参考模型 float {0.1111	相关参数,如 M=1; mask 是滤
float {{0.1111}	板。支持 RAW 图像。
【{0.1111	版:
{0.1111   void   RawImageInversion(char* input, char* output, int width, int height)   (height)	mask[3][3] =
{0.1111   void   RawImageInversion(char* input, char* output, int width, int height)   (height)	11, 0. 1111, 0. 1111},
Void RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤影 input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤影 output, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)	,
Void RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤影 input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤影 output, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)	1, 0. 1111, 0. 1111},
woid RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)  RAW 图像文件名 height 像文件名 height 像文件名 height 像。  RAW 直方 是输入图 中值滤剂    void RAWMedianFilter(char* input, char* output 的行,C    关参数,    参考模材    int    sequence ;  void RawtoBmp1(char* input, char* output, unsigned long Width,    unsigned long Height)  Width 系	, ., ., .,
woid RawImageInversion(char* input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)  RAW 图像文件名 height 像文件名 height 像文件名 height 像。  RAW 直方 是输入图 中值滤剂    void RAWMedianFilter(char* input, char* output 的行,C    关参数,    参考模材    int    sequence ;  void RawtoBmp1(char* input, char* output, unsigned long Width,    unsigned long Height)  Width 系	1, 0. 1111, 0. 1111}};
input, char* output, int width, int height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)  void RAW 图像  AW 图像  AW 图像  AN MER  AN MER  AN MER  AN MER  AN MER  AN MER  AN AN MER  Butter An Mer  Butter An An An An Mer  Butter An An An Mer  Butter An An An An Mer  Butter An An An An Mer  Butter An An An An An An Mer  Butter An An An An An An An An	相,input 是输入的 RAW 图像
height)  void RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmpl(char* input, char* output, unsigned long Width, unsigned long Height)  Width 和  ### A, wi  ### Apple	output 是输出的 RAW 图像文
RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)  Width 系	idth 是输入图像的宽, height
roid RawHistogramEqualization(char* input, char* output, int width, int height)  roid RAWHistogramEqualization(char* input, char* output, int width, int height)  roid RAWMedianFilter(char* input, char* output, int ROWS, int coutput, char* output, int ROWS, int coutput COLS, int M, int sequence[9])  roid RawtoBmp1(char* input, char* output, unsigned long Width, width, width, for the coutput, unsigned long Width,	图像的高。支持 RAW 图像。
RawHistogramEqualization(char* input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)  RAW 直方 是输入图 中值滤镜 output 的行,CO关参数,参考模材 int sequence [9])	均衡化,input 是输入的 RAW
input, char* output, int width, int height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Height)  RAW 图像  A W 图像  A W 图像  A W 图像  A 文件名  Width 系  Width 系	件名,output 是输出的 RAW 图
height)  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤器 input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)	名,width 是输入图像的宽,
像。  void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤剂 input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmpl(char* input, char* output, sequence coutput, unsigned long Width, unsigned long Height)  RAW 直方 是输入图 中值滤剂 output 的行,CO 关参数, 参考模构 int sequence ;  void RawtoBmpl(char* input, char* RAW 图像 output, unsigned long Width, Width 系	
void RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤器 input, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmp1(char* input, char* output, unsigned long Width, unsigned long Height)  RAW 直方 是输入图 中值滤器 output 的行,Control char* input, char* RAW 图像 output, unsigned long Width, Width 系	是输入图像的高。支持 RAW 图
RAWHistogramEqualization(char* input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤剂 output, char* output, int ROWS, int COLS, int M, int sequence[9])  void RawtoBmpl(char* input, char* sequence;  void RawtoBmpl(char* input, char* RAW 图像 output, unsigned long Width, Unsigned long Height)  Width 系	它图物海孙 …····································
input, char* output, int width, int height)  void RAWMedianFilter(char* 中值滤器 input, char* output, int ROWS, int COLS, int M, int sequence[9])  的行,C文参数,参考模构 int sequence;  void RawtoBmp1(char* input, char* output, unsigned long Width, 以文件名 unsigned long Height)  Width 系	方图均衡化,width 和 height
height) void RAWMedianFilter(char* 中值滤器 input, char* output, int ROWS, int COLS, int M, int sequence[9]) 的行,C关参数,参考模型 int sequence; void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width, 以文件和 unsigned long Height)	<b>图像即见和</b> 同。
void RAWMedianFilter(char* 中值滤器 input, char* output, int ROWS, int COLS, int M, int sequence[9]) 的行,C关参数,参考模型 int sequence; void RawtoBmp1(char* input, char* output, unsigned long Width, 以文件器 unsigned long Height)	
input, char* output, int ROWS, int COLS, int M, int sequence[9]) 的行,C 关参数,参考模材 int sequence; void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width, 以文件和 unsigned long Height)	冲 : 目 t 入 ) 文 件 与
COLS, int M, int sequence[9])  的行,C 关参数, 参考模材 int sequence;  void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width, unsigned long Height)  Width 系	波, input 是输入文件名,
关参数, 参考模材 int sequenc ; void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width, unsigned long Height) Width 和	是输出文件名。ROWS 是图像
参考模型 int sequency; void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width, 以文件和 unsigned long Height) Width 系	COLS 是图像的列,M 是滤波相
int sequenc ; void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width,	,如 M=1。支持 RAW 图像。
sequence; void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width, unsigned long Height) Width 系	板:
; void RawtoBmp1(char* input, char* RAW 图像 output, unsigned long Width, unsigned long Height)  Width 系	
output, unsigned long Width, 入文件名 unsigned long Height) Width 系	$ce[9] = \{0, 0, 0, 0, 0, 0, 0, 0, 0\}$
output, unsigned long Width, 入文件名 unsigned long Height) Width 系	
unsigned long Height) Width 禾	象转为 BMP 图像, input 是输
	名,output 是输出文件名。
	和 Height 是输入文件的宽和
□。	
void RawToBmp(char* input, char* RAW 图像	象转为 BMP 图像, input 是输
output, int imageWidth, int 入文件名	名, output 是输出文件名。支
void RawToBmp(char* input,char* RAW 图像	和 Height 是输入文件的宽和 象转为 BMP 图像, input 是输

imageHigth)	<b>持</b> 實和亨和 <b>学</b> 的图 <b>确</b>
	持宽和高相等的图像。
void RGBtoHSI(char* input, char*	RGB 模型转为 HIS 模型,input 是输
output)	入文件名,output 是输出文件名。支
	持 24 位 BMP 图像。
void	拉普拉斯锐化滤波器, input 是输入
RAWLaplacialSharpeningFilter(char	文件名,output 是输出文件名。ROWS
* input, char* output, int ROWS, int	是图像的行大小,COLS 是图像的列大
COLS, int M, float w, float	小, M 和 w 是滤波相关参数, 如 M=1,
mask[3][3])	w=1; mask 是滤波器模板。支持 RAW
	图像。
	参考模板:
	float mask[3][3] = $\{\{0, 1, 0\},$
	$\{1, -4, 1\},$
	$\{0, 1, 0\}\};$
<pre>void RawLaplacianEnhancement(char*</pre>	拉普拉斯算子增强, input1 是输入的
input1, char* output1, int width, int	RAW 图像文件名, output1 是输出的
height)	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 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 Transfer(char* input, char*	传递函数。
output, int width, int height)	
void Homography(char* input1,char*	单应。
input2, char* input3, char*	
output, int width, int height, int	
newwidth, int newheight)	
<pre>void MovieEffect(char* input, char*</pre>	电影效果。
output, int width, int height)	
<pre>void FixedThresholdMethod(char*</pre>	抖色处理,固定阈值法。
input, char* output, int width, int	
height)	
void RandomThresholdMethod(char*	抖色处理,随机阈值法。
input, char* output, int width, int	
height)	
void DitherMatrixMethod(char*	抖色处理,抖动矩阵法,默认 N=2。
input, char* output, int width, int	
height, int N)	
void NormalizedLogBuffer1(char*	对数变换,规范化对数。
input, char* output, int width, int	

hoight)	
height)	<b>动业态格</b> 加芒化动物
void NormalizedLogBuffer2(char*	对数变换,规范化对数。
input, char* output, int width, int	
height)	
void TernaryGrayLevel1(char*	三值灰度。
input, char* output, int width, int	
height)	
void TernaryGrayLevel2(char*	三值灰度。
input, char* output, int width, int	
height)	
void BestEdgeMap1(char*	最佳边贴图。
input, char* output, int width, int	
height)	
void BestEdgeMap2(char*	最佳边贴图。
input, char* output, int width, int	
height)	
void Skeletonize(char* input, char*	骨架化。
output, int width, int height)	14,514,03
void SeparableDiffusion(char*	可分离扩散。
input, char* output, int width, int	474 144 1470
height)	
void Denoising(char* input1, char*	   去除噪声。
input2, char* output, int width, int	
height)	
void Luminosity(char* input, char*	
output, int width, int height)	元文 - 元 - 元 - 元 - 元 - 元 - 元 - 元 - 元 - 元
void Average (char* input, char*	平均化。
output, int width, int height)	
void MinMax(char* input, char*	最小与最大。
output, int width, int height)	取小一取八。 
void Shrink(char* input, char*	   收缩。
1	4又4日。
output, int width, int height)	
void BilinearTransformation(char*	双线性变换。
input, char* output, int width, int	
height, int newwidth, int newheight)	TITI /III / III /
void DitherMatrixMethod(char*	四级抖动,默认 N=2。
input, char* output, int width, int	
height, int N)	
void Dewarped1(char* input, char*	脱蜡。a 是在输出图像中检查半径是
output, int width, int height, int	否<=a, 然后再进行扭曲, 参考:
Offset, double a, double b)	Offset=256, a=256.5, b=0.5。
void Dewarped2(char* input, char*	脱蜡。a 是在输出图像中检查半径是
output, int width, int height, int	否<=a, 然后再进行扭曲, 参考:
Offset, double a, double b, double	Offset=256, a=256.5, b=0.5。

<pre>coeffx[12], double coeffy[12])</pre>	脱蜡规范:
occirr[i2], adabie occir; [i2]	double coeffx[12] =
	{ 1.00056776e+00, -5.68880703e-
	04, -1.13998357e-03,
	1. 00056888e+00, -
	5. 65549579e-04, -1. 13554790e-03,
	9. 99434446e-01 ,
	5. 66658513e-04 , 1. 13110351e-
	03,
	9. 99433341e-01 ,
	5. 67767429e-04 , 1. 13553921e-
	03 };
	05 ],
	double coeffy[12] = {-
	5. 67763072e-04, 1. 00056888e+00,
	1. 13998357e-03, 5. 68880703e-04,
	9. 99434450e-01, -1. 13554790e-03,
	5. 65553919e-04,
	· · · · · · · · · · · · · · · · · · ·
	9. 99433341e-01, -1. 13110351e-03,
	-5. 66658513e-04,
	1. 00056777e+00, 1. 13553921e-
	03}; 於理八年山 [ M42] [ V_C
void TextureSegmentation1(char*	纹理分割,默认 K=6, N=100。
input, char* output, int width, int	
height, int K, int N)	を対す用 八字山 - 関4 2.1 - V − C - N − 1 0 0
void TextureSegmentation2(char*	纹理分割,默认 K=6, N=100。
input, char* output, int width, int	
height, int K, int N)	· · · · · · · · · · · · · · · · · · ·
void TextureClassification(vector	纹理分类,a 是要分类的图像的数量,
<pre><string> filename, char* output, int</string></pre>	如 filename 里有 3 个图像名称,则
width, int height, int K, int N, int	a=3; output 是分类结果文件,格式
a)	为 txt 的文本文件; 默认 K=4, N=1000。
void ErrorDiffusion1(char*	误差扩散。
input, char* output, int width, int	
height)	口去护忱
void ErrorDiffusion2(char*	误差扩散。
input, char* output, int width, int	
height)	) 17 24 12 H/.
void ErrorDiffusion3(char*	误差扩散。
input, char* output, int width, int	
height)	TT 16 1 . 11
void Thin(char* input, char*	图像细化。
output, int width, int height)	

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

# BMP 图像处理

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

	14.11.14.1.77.14
	像的 Alpha 通道
	BMPMat;
unsigned int BMPHeight(char*	读取 BMP 图像的高度。
input)	No. 7
unsigned int BMPWidth(char*	读取 BMP 图像的宽度。
input)	
void GenerateImage(char*	生成 24 位和 32 位 BMP 图像。type 等于图
output, BMPMat**	像的位数,如 type=24。
color, unsigned short type)	参考用例:
	BMPMat** color =
	(BMPMat**) malloc(sizeof(BMPMat*)*1280
	);
	for (unsigned int $i = 0$ ; $i < 1280$ ;
	i++)
	{
	color[i] =
	(BMPMat*) malloc(sizeof(BMPMat)*2450);
	}
	for (unsigned int $i = 0$ ; $i < 1280$ ;
	i++) {
	for (unsigned int j = 0; j <
	2450; j++) {
	color[i][j].B =0;
	color[i][j].G =0;
	color[i][j].R =255;
	}
	}
void	直方图均衡,支持8位和16位BMP。input
HistogramEqualization5(char*	是输入文件名,output 是输出文件名。
input, char* output)	
void Resize(char*	图片缩放,支持8位和16位BMP。input是
input, char* output, int	输入文件名, output 是输出文件名。Height
Height, int Width)	和 Width 是输出图像的高和宽。
double MeanBrightness(char*	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* input, char* output, char yuvmode)	BMP 图像转为 YUV 图像, input 是输入文件 名, output 是输出文件名。yuvmode 是 YUV 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*	Ins1977 滤镜, input 是输入文件名, output
input, char* output, int	是输出文件名。参考: ratio=100。
ratio)	
void LOMO(char* input, char*	LOMO 滤镜,DarkAngleInput 是暗角模板图
DarkAngleInput, char*	像名,参考: ratio=100。
output, int ratio)	
void PNGGray(char*	图像灰度化, input 是输入文件名, output
input, char* output)	是输出文件名。
void PNGSpotlight(char*	聚光灯效果, input 是输入文件名, output
input, char* output, int	是输出文件名。焦点坐标
centerX, int centerY, double	( centerX,centerY ) , 如 :
a, double b, double c, double	centerX=400, centerY=180; a, b, c, d, e
d, double e)	是相关参数,默认 a=100,b=100,c=160,
	d=80, e=0.5 <sub>°</sub>
void PNGIllinify(char*	幻化效果, input 是输入文件名, output 是
input, char* output)	输出文件名。
void PNGWaterMark(char*	图像加水印, input1 和 input2 的尺寸必须
input1, char* input2, char*	相同。
output)	
void Short(char* input, char*	矮化特效。a=1, b=128, c=2, d=0.5,
output, int a, int b, int	depth=24。支持 24 位 BMP 图像。
c, double d, int depth)	
void Rise(char* input, char*	增高特效。a=1, b=128, c=0.5, d=2,
output, int a, int b, double	depth=24。支持 24 位 BMP 图像。
c, int d, int depth)	To a state of the
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)	) - 174.
void Fat (char* input, char*	肥胖特效。a=1,b=128,c=0.5,depth=24。
output, int a, int b, double	支持 24 位 BMP 图像。
c, int depth)	7.11 = 1 = 1 = 1 M. o
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)	depending 214 bit E bit Edward
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	dopon 210 XII E Dim Edition
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)	2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
void Winding (char*	弯曲特效。1im=20, a=1, b=128, c=4, d=5,
input, char* output, int	e=0.5, depth=24。支持 24 位 BMP 图像。
lim, int a, int b, int c, int	, , , , , , , , , , , , , , , , , , ,
d, double e, int depth)	
void ImageFusion3(char*	多聚焦图像的融合,支持 8 位 BMP 图像。
input1, char* input2, char*	block_height=8 , block_width=8 ,
output, int block_height, int	threshold=1.75.
block_width, double	
threshold)	
void ImageFusion4(char*	多聚焦图像的融合,支持 8 位 BMP 图像。
input1, char* input2, char*	block_height=8 , block_width=8 ,
output, int block_height, int	threshold=1.75。
block_width, double	
threshold)	
void ImageFusion5(char*	图像融合。参考: a=3, b1=4, DX1=-68, DY1=-
input1, char* input2, char*	99, EPS=1, input1="图像融合 1.jpg",
MaskImage, char* output, int	input2="图像融合 2.jpg", MaskImage="掩
dx[], int dy[], int a, double	膜.png", output="output.jpg"。
b1, int DX1, int DY1, double	int $dx[] = \{0, 0, -1, 1\};$
EPS)	int $dy[] = \{-1, 1, 0, 0\};$
void Screenshot3(HWND hWnd,	截屏函数。hWnd 是要截屏的窗口句柄,如:
LPCWSTR OutputImage)	GetDesktopWindow();OutputImage 截图名
	称。
void Screenshot1(HWND hWnd,	截屏函数。hWnd 是要截屏的窗口句柄,如:
LPCWSTR OutputImage)	GetDesktopWindow();OutputImage 截图名
	称。
void Screenshot2(HWND	截屏函数。hWnd 是要截屏的窗口句柄,如:
hWnd, LPCWSTR OutputImage)	GetDesktopWindow();OutputImage 截图名

	称。
void Dark(char* input, char*	暗调滤镜,参考: ratio=100。
output, int ratio)	HI MINE DE , 多可, I dt I O 100。
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
degree, int d)	生成 TGA 图像,参考: degree=10。
void PinchFilter(char*	挤压形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int a)	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
imput, onar · output, int a)	a=3 时生成 TGA 图像。
void PinchFilter(char*	挤压形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
	坐标, cenY 是形变中心点Y 坐标。
void SpherizeFilter(char*	球面形变特效滤镜,a=0时生成BMP图像,
input, char* output, int a)	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
	a=3 时生成 TGA 图像。
void SpherizeFilter(char*	球面形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像, a=2 时生成 PNG 图像,
cenX, int cenY, int a)	a=3 时生成 TGA 图像, cenX 是形变中心点 X
, , , ,	坐标, cenY 是形变中心点 Y 坐标。
void SwirlFilter(char*	旋转形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像,a=2 时生成 PNG 图像,
ratio, int a)	a=3 时生成 TGA 图像,ratio=3。
void SwirlFilter(char*	旋转形变特效滤镜, a=0 时生成 BMP 图像,
input, char* output, int	a=1 时生成 JPG 图像,a=2 时生成 PNG 图像,
cenX, int cenY, int ratio, int	a=3 时生成 TGA 图像,ratio=3,cenX 是形
a)	变中心点 X 坐标,cenY 是形变中心点 Y 坐
	标。
void ClosedOperation(char*	闭运算, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 4 位 BMP 图像。
void ColorTransfer(char*	颜色转移。
input1, char* input2, char*	
output)	
void GrayImagel(char*	直方图均衡化。
input, char* output)	
void ChannelHisteq(char*	直方图均衡化。
input, char* output)	and the Till a gar
void HSVtoRGB(char* input,	HSV 转到 RGB。
char* output)	+->-FII-1/1-11.
void	直方图均衡化。
HistogramEqualizationOnGrayI	
mage(string input, char*	
output)	

CImg <unsigned int=""></unsigned>	直方图均衡化。
HistogramEqualizationOnGrayI	五万日 <b>7</b> 6万円。
mage2(string input)	
void	直方图均衡化。
HistEqualColorImageOneColorC	五万国为 <b>员</b> 化。
hannel(string input, char*	
output)	
CImg <unsigned int=""></unsigned>	直方图均衡化。
HistEqualColorImageOneColorC	五万国为民化。 
hannel1(string input)	
void	直方图均衡化。
HistEqualColorImageThreeColo	五万日 <b>岁</b> 族化。
rChannels(string input,	
char* output)	
CImg <unsigned int=""></unsigned>	直方图均衡化。
HistEqualColorImageThreeColo	五万日 <b>岁</b> 族化。
rChannels(string input)	
void	HSI 空间。
HistEqualColorImageHSISpace(	1101 1.141.
string input, char* output)	
CImg <unsigned int=""></unsigned>	HSI 空间。
HistEqualColorImageHSISpace(	1101 1110
string input)	
void ColorTransfer1(char*	颜色转移。
sourceImage, string	
targetImage, char* output)	
CImg <unsigned int=""></unsigned>	颜色转移。
ColorTransfer2(string	
sourceImage, string	
targetImage)	
void BMPtoJPG(char*	BMP 图像转为 JPG 图像。支持 24 位 BMP 图
input, char* output, int a)	像,尺寸必须是 8 的倍数, a 代表文件压缩
input, onar · output, int a)	程度,数字越大,压缩后的文件体积越小,
	如 a=100。
void	部分颜色保留滤镜,参考: ratio=60。
PartialColorRetention(char*	HEAD WOLLD A 1. TOOLO OOL
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	
void	彩色图转灰度图, input 是要处理的彩色图

Crow I maga Convention (shows	クロス
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 是初始阈
	值,如 BeforeThreshold=10。支持 8 位 BMP
	图像。
void LowerBrightness(char*	调低亮度, input 是输入文件名, output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100,b=0。
void HightBrightness(char*	调高亮度, input 是输入文件名, output 是
input, char* output, int a, int	输出文件名。支持 24 位 BMP 图像。a 和 b
b)	的参考值可为 a=100, b=0。
void	迭代阈值选择, input 是输入文件名,
IterativeThresholdSelection(	output 是输出文件名。支持 8 位 BMP 图像。
char* input, char* output)	
void DitheringMethod(char*	抖动法, input 是输入文件名, output 是输
input, char* output)	出文件名。支持8位BMP图像。
void LogTransformation(char*	对数变换, input 是输入文件名, output 是
input, char* output, int	输出文件名。支持 8 位 BMP 图像。constant
constant)	是相关参数,如 constant=15。
void	对数变换, input 是输入文件名, output 是
LogarithmicTransformation(ch	输出文件名。支持 BMP 图像。
ar* input, char* output)	他们人们有。 文內 Dimi 国家。
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)	Output 定制田文川石。文诗 DMI 国家。
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)	
· · · · · · · · · · · · · · · · · · ·	
void Expansion(char*	二值图像膨胀,参考:
input, char* output, unsigned	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
char mask[9], int c)	, c=128。 一 店 図 焔 麻 畑
void Corrosion(char*	二值图像腐蚀,参考:
input, char* output, unsigned	$\max \{9\} = \{0, 255, 0, 255, 255, 255, 0, 255, 0\}$
char mask[9], int c)	, c=128。
void OpenOperation(char*	二值图像开运算,参考:
input, char* output, unsigned	$mask[9] = \{0, 255, 0, 255, 255, 255, 0, 255, 0\}$
char mask[9], int c)	,c=128。 一 唐 图 梅 印 二 数
void ClosedOperation(char*	二值图像闭运算,参考:
input, char* output, unsigned	$\max [9] = \{0, 255, 0, 255, 255, 255, 0, 255, 0\}$
char mask[9], int c)	, c=128。
void	二值图像开运算提取轮廓,参考:
OpenOperationToExtractContou	$mask[9] = \{0, 255, 0, 255, 255, 255, 0, 255, 0\}$
r(char* input, char*	, c=128°

output, unsigned char	
mask[9], int c)	
void	二值图像膨胀运算提取轮廓,参考:
ExpansionOperationToContourE	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
xtraction(char* input, char*	, c=128.
output, unsigned char	
mask[9], int c)	
void	二值图像腐蚀运算提取轮廓,参考:
CorrosionCalculationToContou	mask[9]={0, 255, 0, 255, 255, 255, 0, 255, 0}
rExtraction(char*	, c=128。
input, char* output, unsigned	
char mask[9], int c)	
void Glaw(char* input, char*	发光滤镜,参考: ratio=100。
output, int ratio)	
void LowPassFilter(char*	低通滤波器,input 是输入文件名,output
input, char* output)	是输出文件名。支持 BMP 图像。
void HighPassFilter(char*	高通滤波器,input 是输入文件名,output
input, char* output)	是输出文件名。支持 BMP 图像。
void Thinning(char*	图像细化, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 BMP 图像。
void ThinningLine(char*	图像细化且线条化,input 是输入文件名,
input, char* output)	output 是输出文件名。支持 BMP 图像。
void Corrosion(char*	腐蚀, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 4 位 BMP 图像。
void Corrosion1(char*	腐蚀, input 是输入文件名, output 是输出
input, char* output, int	文件名。支持 24 位 BMP 图像。TempBuf 是
*TempBuf, int TempH, int	腐蚀模板,TempH 和 TempW 分别是 TempBuf
TempW)	的高和宽,如 TempH=4,TempW=4,则有
	TempBuf[4][4].
void Expand(char*	膨胀, input 是输入文件名, output 是输出
input, char* output, int	文件名。支持 24 位 BMP 图像。TempBuf 是
*TempBuf, int TempH, int	膨胀模板,TempH 和 TempW 分别是 TempBuf
TempW)	的高和宽,如 TempH=4,TempW=4,则有
	TempBuf[4][4]。
void	高斯滤波,支持 PNG 文件。
GaussianBlurFilter(char*	
input, char* output)	
void GaussianFiltering(char*	高斯滤波, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void	拉普拉斯增强, input 是输入文件名,
LaplaceEnhancement(char*	output 是输出文件名。支持 24 位 BMP 图
input, char* output)	像。
void Residual(char*	求残差, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。

i-1 C1: -1+F: 1+/-1	火四柱分准淬 :
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 <sub>°</sub>
void Compress(char*	压缩,支持多种文件。input 是要压缩的文
input, char* output)	件名, output 是压缩后的文件名。
void Decompression(char*	解压缩,支持多种文件。input 是要解压缩
input, char* output)	的文件名,output 是解压缩后的文件名。
void BlackWhite(char*	黑白化, input 是输入的原图像, output 是
input, char* output)	输出的黑白图像。支持24位BMP图像。
void Underexposure(char*	图像欠曝光, input 是输入的原图像,
input, char* output)	output 是输出的欠曝光图像。支持 24 位
input, char* output)	
	BMP 图像。
void Overexposure(char*	图像过曝光, input 是输入的原图像,
input, char* output)	output 是输出的过曝光图像。支持 24 位
	BMP 图像。
void Nostalgia(char*	怀旧滤镜, input 和 Mask 都是输入的文件
input, char* Mask, char*	名,Mask 是褶皱图像路径,ratio=100。
output, int ratio)	
void GammaTransform(char*	伽马变换, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void GrayScale(char*	灰度化, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void	灰度图二值化, bit 用于设定位数, 如
GrayImageBinarization(char*	bit=8; threshold 是阈值,如
input, char* output, int	threshold=200。支持8位BMP图像。
bit, int threshold)	on concrete to the part of the
void GreyPesudoColor(char*	灰度图伪彩色化, input 是输入文件名,
-	output 是输出文件名。支持 8 位 BMP 图像。
input, char* 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)	Elle Este de La Companya de la Compa
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
	/44/ <del>                                     </del>
	切系数,负值是向左 / 向下偏移。支持 BMP

	图像。
void Scaled(string	缩放操作, input 是输入的文件, output 是
input, char* output, double	缩放操作,filput 定栅入的文件,output 定   缩放操作后的结果文件名, cx 和 cy 分别是
	横向及纵向的缩放系数,系数大于1表示拉
cx, double cy)	
Deteted 1 (atrice	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 MeanFltering1(char*	均值滤波,input 是输入文件名,output 是
input, char* output)	输出文件名。支持 8 位和 24 位 BMP 图像。
void KapoorAlgorithm(char*	卡普尔算法,input 是输入文件名,output
input, char* output, int	是输出文件名。BeforeThreshold是初始阈
BeforeThreshold)	值,如 BeforeThreshold=150。支持 8 位 BMP
	图像。
void OpenOperation(char*	开运算, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 4 位 BMP 图像。
void Diffusion(char*	扩散滤镜,参考: ratio=90。
input, char* output, int	
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
kernel[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,

	100
BrightnessAdjustment1(char*	contrast=100.
input, char* output, int	
brightness, int contrast)	
void	亮度对比度调节,参考: brightness=-30,
BrightnessAdjustment2(char*	contrast=100°
input, char* output, int	
brightness, int contrast)	
void	零填充与对称扩展,支持8位和24位BMP
ZeroFillingSymmetricExtensio	图像。
n(char* input, char* output)	
void PopArtStyle(char*	流行艺术风滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void LightLeakage(char*	漏光滤镜, input 和 Mask 都是输入的图像
input, char* Mask, char*	名,Mask 是漏光模板图像,ratio=90。
• '	石, Mask 足M 几天似日 家, 1 at 10 500
output, int ratio)	好你读过。"······· 目於入文件与
void LinearFiltering(char*	线性滤波, input 是输入文件名, output 是
input, char* output, short	输出文件名。支持 8 位 BMP 图像。
average[3][3])	参考模板:
	short average[3][3] = $\{\{1, 2, 1\},$
	$\{2, 4, 2\},\$
	$\{1, 2, 1\}\};$
void MedianFiltering(char*	中值滤波, input 是输入文件名, output 是
input, char* output, short	输出文件名。支持 8 位 BMP 图像。
average[3][3])	参考模板:
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\},\$
sob1e2[3][3])	$\{1, 2, 1\}\};$
	short soble1[3][3] = $\{\{-1, -2, -1\},$

	$\{0, 0, 0\},\$
	$\{1, 2, 1\}\};$
	short $soble2[3][3] = \{\{-1, 0, 1\},\$
	$\{-2, 0, 2\},\$
	$\{-1, 0, 1\}\};$
void	算术平均滤波器, input 是输入文件名,
ArithmeticMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
void	几何平均滤波器, input 是输入文件名,
GeometricMeanFilter(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	
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 Mosaic(char*	马赛克化图像, input 是输入文件名,
input, char* output, int x)	output 是输出文件名。x 是马赛克处理的块
	的大小。支持 24 位 BMP 图像。
void MosaicFilter(char*	马赛克滤镜,参考: ratio=50。
input, char* output, int	
ratio)	
void Expansion(char*	膨胀, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 4 位 BMP 图像。
void MidSmoothing(char*	中值滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void AvgSmoothing(char*	均值滤波器, input 是输入文件名, output
input, char* output)	是输出文件名。支持 8 位 BMP 图像。
void Averaging(char*	图像平均化, input 是输入文件名, output
input1, char* input2, char*	是输出文件名。a 是平均化相关参数,如
input3, char* output, int a)	a=3。支持 8 位 BMP 图像。
void PlaneSlicing(char*	平面切片, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void Translation(char*	图像平移,参考: xoffset=-100, yoffset=-
input, char* output, int	100。
xoffset, int yoffset)	
void	锐化空间滤波器, input 是输入文件名,
SharpeningSpatialFiltering8(	output 是输出文件名。model 是锐化模板。
char* input, char* output, int	支持8位灰度图像。
mode1[9])	
void   PseudoGrayscale(char*	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。
	二色化,input 是输入文件名,output 是输
void TwoColors (char*	出文件名。 threshold 是阈值,如
input, char* output, int	
threshold, unsigned char	threshold=115; color1 和 color2 是要填
color1, unsigned char color2)	充的两个颜色。支持 24 位 BMP 图像。
void	filename 是生成的 PNG 图像文件名; img 是
PNGImageGeneration(char*	图像的像素数据,W是图像的宽,H是图像
filename, const unsigned char	的高, x=0 选择生成 RGB 图像, x=1 选择生
img[], unsigned W, unsigned	成 RGBA 图像。
H, int x)	
void MakeSphere(double	使用反射模型在正交投影下生成球体的图
V[3], double S[3], double r,	像,V是摄影机的方向,output 是输出的结
double a, double m, int ROWS,	果图像文件名,ROWS 是输出图像的行数,
int COLS, char* output)	COLS 是输出图像的列数,参考: V[3] =
	$\{0.0, 0.0, 1.0\}, S[3] = \{0.0, 0.0, 1.0\}$
	1.0}, r=50, a=0.5, m=1。支持 RAS 文件。
void MakeSphere(double	使用反射模型生成球体的图像,vector_v
vector_v[3], double	是摄影机的方向,output 是输出的结果图
vector_s[3], double r, double	像文件名,ROWS 是输出图像的行数,COLS
a, double m, int ROWS, int	是输出图像的列数,参考: vector_v[3] =
COLS, char* output, double	$\{0.0, 0.0, 1.0\}, \text{ vector\_s}[3] = \{0.0, \dots \}$
max)	0.0, 1.0}, r=50, a=0.5, m=1。支持 RAS 文
	件。
void	双边滤波, input 是输入文件名, output 是
BilateralFiltering(string	输出文件名。支持 24 位 BMP 图像。ssd 和
input, char* output, double	sdid 分别是空间域标准差与强度域标准
ssd, double sdid)	差。
void	具有圆形结构集的双层形态侵蚀,支持8位
DoubleLayerErosion(char*	和 24 位 BMP 图像。
input, char* output)	古界内域 台州 100
void SketchFilter(char*	素描滤镜,参考: ratio=100。
input, char* output, int	
ratio)	<i>院</i> 计 1 V F 1 V F
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)	医加克斯思士 ·
void AddGaussNoise(char*	添加高斯噪声, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
void	添加椒盐噪声, input 是输入文件名,
AddSaltPepperNoise(char*	output 是输出文件名。支持 8 位 BMP 图像。
input, char* output)	(国送八) 文 ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( )
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 是绿色通道图像。Part 1946年
Goutput, char* Boutput)	像,Boutput 是绿色通道图像。支持 24 位

	BMP 图像。
Dotterm Method (shere)	图案法, input 是输入文件名, output 是输
void PatternMethod(char*	
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	c=0.0722。支持 24 位 BMP 图像。
a, double b, double c)	
void HighPassFilter(char*	高通滤波器,参考: preserve=0。
input, char* output, int	
preserve)	
void EmbossFilter(char*	浮雕过滤器,参考: preserve=1。
input, char* output, int	
preserve)	
void SharpenFilter(char*	锐化过滤器,参考: preserve=1。
input, char* output, int	
preserve)	
void Convolution(char*	卷积,参考: w=7, preserve=1。
input, char* output, int w, int	
preserve)	
void GaussianBlur(char*	高斯模糊,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	
void HybridImage(char*	混合图像,参考: sigma=2, preserve=1。
input1, char* input2, char*	
output, float sigma, int	
preserve)	
void LowFrequencyImage(char*	低频图像,参考: sigma=2, preserve=1。
input, char* output, float	
sigma, int preserve)	Autoria de la companya della companya della companya de la companya de la companya della company
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage(char*	
input, char* output, float	
sigma, int preserve)	京個圆角 <u>乡</u> 老 · 0
void	高频图像,参考: sigma=2, preserve=1。
HighFrequencyImage1(char*	
input, char* output, float	
sigma, int preserve)	双边滤波 <u> </u>
void Bilateral(char*	双边滤波,参考: sigma1=3, sigma2=0.1。
input, char* output, float	
sigma1, 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	皮肤细屑,a 定于用级别,b 代表定省应用 皮肤过滤器,a=2,b=1。
b)	/文川八人土が台前
void Resizel(char*	■ 图像模糊,w=713,h=467。
voiu kesizei (char*	□ ⑤ ③ (天'彻, W-110, II-401。

· · · · · · · · · · · · · · · · · · ·	
input, char* output, int w, int	
h)	TTT Mr. Idda.ldg
void Resize2(char*	图像模糊。
input, char* output, int w, int	
h)	
void Shift(char* input, char*	Shift函数, ch=1, v=0.1。
output, int ch, float v)	
void RGBtoHSV(char*	RGB 转 HSV。
input, char* output)	
void HSVtoRGB(char*	HSV 转 RGB。
input, char* output)	
void RGBtoLCH(char*	RGB 转 LCH。
input, char* output)	
void LCHtoRGB(char*	LCH 转 RGB。
input, char* output)	
void ColorTransfer(char*	颜色转移。
input1, char* input2, char*	
output)	
void DrawText(char*	文本绘制,R=255,G=255,B=255,depth=1,
<pre>inputText, char* output, int</pre>	spectrum=3, (x,y)是文本的坐标, color1
width, int height, int	是前景颜色, color2 是背景颜色,
depth, int spectrum, int x, int	opacity=1, font=60.
y, unsigned char R, unsigned	
char G, unsigned char	
B, unsigned char	
color1[], unsigned char	
color2[], float	
opacity, unsigned int font)	
void EqualizedGray(char*	灰度图直方图均衡化。
input, char* output)	
void	彩色图直方图均衡化。
ColorHistogramEqualization(c	
har* input, char* output)	
void AverageHistogram(char*	直方图均衡化。
input, char* output)	
void HSIHist(char* input,	HIS 直方图。
char* output)	,
void ImageCutting(char*	图像裁剪, input 是输入文件名, output 是
input, char* output, int	输出文件名。leftdownx, leftdowny,
leftdownx, int leftdowny, int	rightupx, rightupy 是要裁剪的矩形区域
rightupx, int rightupy)	的左下角和右上角的坐标(连续四个整数
	值,如 50 50 300 300)。支持 24 位 BMP 图
	像。
void	图像层算法。
L	/ = / 2 /

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
	图像。
void	分段线性变换, input 是输入文件名,
PiecewiseLinearTransform(cha	output 是输出文件名。支持 8 位 BMP 图像。
r* input, char* output)	
void PowerConvertion(char*	功率转换,input 是输入文件名,output 是
input, char* output, double	输出文件名。如: c = 1.2, g = 0.5。支持
c, double g)	8 位 BMP 图像。
void Smooth(char*	平滑, input 是输入文件名, output 是输出
input, char* output)	文件名。支持 8 位 BMP 图像。
void Multiply(char*	图像倍增化,input 是输入文件名,output
input, char* output, int N, int	是输出文件名。如: N=1。支持 8 位 BMP 图
Sb1Mask1[3][3], int	像。
Sb1Mask2[3][3], int	参考模板:
Lap1Mask[3][3])	int Lap1Mask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
	int Sb1Mask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
,	};
void Add(char* input, char*	图像添加, input 是输入文件名, output 是
output, int N, int	输出文件名。如: N=1。支持 8 位 BMP 图像。
Sb1Mask1[3][3], int	参考模板:
Sb1Mask2[3][3], int	int Lap1Mask[3][3] = {
LaplMask[3][3])	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
	int Sb1Mask1[3][3] = {

	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
	};
void PowerConvertion1(char*	功率变换, 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, N=1。
c, double g, int N, int	支持 8 位 BMP 图像。
Sb1Mask1[3][3], int	int Lap1Mask[3][3] = {
Sb1Mask2[3][3], int	0, 1, 0,
Lap1Mask[3][3])	1, -4, 1,
	0, 1, 0
	};
	int Sb1Mask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
	};
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,
	coefficient=16。支持 24 位 BMP 图像。
Template[3][3], int	
coefficient)	
· · · · · · · · · · · · · · · · · · ·	平滑, input 是输入文件名, output 是输出
	文件名。Template 是平滑模板,均一化处
- '	理, coefficient1 = 9。支持24位BMP图
/	- 111
	像。
	参考: a=0.33, b=0.33, c=0.33。支持 24 位
	BMP 图像。
holding(const LPCTSTR input,	
const LPCTSTR output, double	
a, double b, double c)	
void AdjustPixel(char*	调整像素值, input 是输入文件名, output
_	调整像素值, input 是输入文件名, output 是输出文件名。a 是用于设置图像像素的相

```
彩色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,
ColorRectangleLocalSegmentat
ion(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
x2, int y2, BMPMat color)
                              需引入以下头文件:
                              typedef struct {
                                  unsigned char B;
                                  unsigned char G;
                                  unsigned char R;
                                  unsigned char A;
                              }BMPMat;
                              声明:
                              unsigned
                                           char**
                                                     BMPRead8(char*
                              input);
                              void
                                               GenerateImage8(char*
                              output, unsigned char** color);
                              BMPMat** BMPRead(char* input);
                              void
                                                GenerateImage(char*
                              output, BMPMat** color, unsigned short
                              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
                              0; j \le dth; j++) {
                                         output[i][j]. B=color. B;
                                         output[i][j]. G=color. G;
                                         output[i][j]. R=color. R;
                                  for (unsigned
                                                    int
                                                            i
                              y1; i = y2; i++  {
```

for (unsigned

int

```
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
GrayRectangleLocalSegmentati
                              充,(x1,y1)是矩形的左上角的坐标,
                              (x2, y2) 是矩形的右下角的坐标。
on (char*
                 input, char*
                              函数源代码:
output, int
             x1, int
                      y1, 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);
                                                             char**
                                 unsigned
                              output=BMPRead8(inputfile);
                                 unsigned
                                                                int
                              height=BMPHeight(inputfile);
                                 unsigned
                                                                int
                              width=BMPWidth(inputfile);
                                  for (unsigned
                                                   int
```

```
0; i < height; i++) {
                                       for (unsigned
                                                       int
                                                              j
                              0; j \le (idth; j++) {
                                         output[i][j]=color;
                                   for (unsigned
                                                             i
                                                    int
                              y1;i = y2;i++) {
                                       for (unsigned
                                                       int
                                                              j
                              x1; j \le x2; j++) {
                                           output[i][j]=input[i][j];
                                  }
                              GenerateImage8(outputfile, output);
                               彩色图画矩形,(x1, y1)是矩形的左上角
void
ColorDrawRectangle(char*
                               的坐标,(x2, y2)是矩形的右下角的坐标。
                              函数源代码:
input, char*
                  output, int
                               需引入以下头文件:
x1, int
           y1, int
                      x2, int
y2, BMPMat 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:
                                  BMPMat color=\{255, 255, 255\};
                                  BMPMat**
                               input=BMPRead(inputfile);
                                  BMPMat**
                              output=BMPRead(inputfile);
                                  unsigned
                                                                  int
                              height=BMPHeight(inputfile);
```

```
unsigned
                                                                       int
                                 width=BMPWidth(inputfile);
                                     for (unsigned
                                                         int
                                                                  i
                                 0; i < height; i++) {
                                          for (unsigned
                                                           int
                                                                   j
                                 0; j \le (idth; j++) 
                                            output[i][j].B=color.B;
                                            output[i][j]. G=color. G;
                                            output[i][j]. R=color. R;
                                     for (unsigned
                                                         int
                                                                  i
                                 0; i \leq height; i++) 
                                          for (unsigned
                                                           int
                                                                   j
                                 0; j \le (idth; j++) {
                                            if(j)=x1&&j<=x2&&i==y1)
                                              output[i][j].B=color.B;
                                            output[i][j]. G=color. G;
                                            output[i][j]. R=color. R;
                                              if(j==x1\&\&i>=y1\&\&i<=y2)
                                              output[i][j].B=color.B;
                                            output[i][j]. G=color. G;
                                            output[i][j]. R=color. R;
                                              if(j==x2\&\&i>=y1\&\&i<=y2)
                                              output[i][j]. B=color. B;
                                            output[i][j]. G=color. G;
                                            output[i][j]. R=color. R;
                                              if(j)=x1\&\&j<=x2\&\&i==y2)
                                              output[i][j].B=color.B;
                                            output[i][j].G=color.G;
                                            output[i][j]. R=color. R;
                                          }
                                     }
                                 GenerateImage (outputfile, output, 24);
                                 灰度图画矩形,(x1,y1)是矩形的左上角
void GrayDrawRectangle(char*
```

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

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

.1	\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
void EdgeDetection(char*	边缘检测,参考: cutoff=115。
input, char* output, double	
cutoff)	197.0
void Sharpen(char*	锐化。
input, char* output)	
void CannyProcessing(char*	Canny 处理, a 可以为 1、2、3、4、5。支持
input, char* output, int a)	BMP 图像。
void AverageGrayScale(char*	平均灰度化。
input, char* output)	
void SimpleBW(char*	简易 BW。
input, char* output)	
void AdvancedBW(char*	高级 BW。
input, char* output)	
void UniformNoise(char*	均匀噪声。
input, char* output)	
void GaussianNoise(char*	高斯噪声。
input, char* output, double	
sigma)	
void	椒盐噪声。
SaltAndPepperNoise(char*	
input, char* output)	
void MeanFilter(char*	均值滤波。
input, char* output, int	
filterSize)	
void GaussianFilter(char*	高斯滤波。
input, char* output, double	
sigma)	
void MedianFilter(char*	中值滤波。
input, char* output, int size)	
void	有效均值滤波器。
EfficientMeanFilter(char*	
input, char* output, int	
filterSize)	
double	均方误差, 计算图像相似度, 返回值越小图
MeanSquaredError(char*	像就越相似。
input1, char* input2, char*	
output)	
void GrayAVS(char*	input 是输入文件名, output 是输出文件
input, char* output, float	名。支持 8 位 BMP 图像。
k, float b)	
void	直方图均衡化, input 是输入文件名,
HistogramEqualize24(char*	output 是输出文件名。支持 24 位 BMP 图
input, char* output)	像。
void	矩阵变换。
	/ HI I / C / / C ·

MatrixTransformation(char*	
input, char* output)	— /= /L
void Binarization(char*	二值化。
input, char* output)	1. <del></del>
void	分离出蓝色通道。
ChannelSeparation_B(char*	
input, char* output)	
void	分离出绿色通道。
ChannelSeparation_G(char*	
input, char* output)	
void	分离出红色通道。
ChannelSeparation_R(char*	
input, char* output)	
void Inverse(char*	反转。
input, char* output)	
void	直方图均衡化。
HistogramEqualization8(char*	
input, char* output)	
void Smooth(char*	平滑。
input, char* output)	
void 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)	
<pre>void GryOppositionSSE(char*</pre>	input 是输入文件名, output 是输出文件
input, char* output)	名。支持8位BMP图像。
void 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。

Translation24(char* input, char* output)   直方图统计,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   直方图均衡化,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   では ReflectionRay(char* input, char* output)   出文件名。支持 24 位 BMP 图像。   反射线,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   反射线,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   「反射线,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   均值滤波,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   中值滤波,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   中值滤波,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   中值滤波,input 是输入文件名,output 是输出文件名。支持 24 位 BMP 图像。   では
Example   BistogramEqualization1 (char* input, char* output)
HistogramEqualization1(char* input, char* output)  void ReflectionRay(char* 反射线, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。  void MeanFiltering24(char* input, char* output)  void MedianFiltering24(char* input, char* output)  void MedianFiltering24(char* input, char* output)  void ZoomOutAndZoomIn(char* input, char* output, double value)  void Translation24(char* input, char* output, int x, int y)  void Mirror24(char* input, char* output, int x, int input, char* output)  void Mirror24(char* input, char* output)  void Mirror24(char* input, char* output)  void Mirror24(char* input, char* output)  void Rotate24(char* input, char* output)  void Rotate24(char* input, char* output, double degree)  presented BMP B(g.  with Ambrev And DMP B(g.  with Ambrev And DMP B(g.  presented BMP
input, char* output
void ReflectionRay(char* input, char* output) void MeanFiltering24(char* input, char* output) void MeanFiltering24(char* 均值滤波, input 是输入文件名, output 是输出文件名。支持 24 位 BMP 图像。 void MedianFiltering24(char* input, char* output) void ZoomOutAndZoomIn(char* input, char* output, double value) void Translation24(char* input, char* output, int x, int y)  void Mirror24(char* input, char* output, int x, int y)  void Rotate24(char* input, char* output) void Rotate24(char* input, char* output) void Rotate24(char* input, char* output, double degree)  void Rotate24(char* input, char* output, double degree)  void Rotate24(char* input, char* output)  void Rotate24(char* input, char* output, double degree)  prefixed BMP 图像。  void Rotate24(char* input, char* output, double degree)  prefixed BMP 图像。  void Simput 是输入文件名, output 是输出文件名。  prefixed BMP 图像。  void BMP 图像。  void Simput 是输入文件名, output 是输出文件名。  prefixed BMP 图像。  void BMP 图像。
input, char* output) void MeanFiltering24(char* input, char* output) widdianFiltering24(char* input, char* output, double output 是输出文件名。 value 是放大倍数,如 value=0.5。 支持 24 位 BMP 图像。  woiddianFiltering24(char* input, char* output, double output 是输出文件名。 value 是放大倍数,如 value=0.5。 支持 24 位 BMP 图像。  woiddianFiltering24(char* input, char* output, int x, int input, char* output, int x, int input, char* output) woiddianFordalianFiltering24(char* input, char* output) woiddianFordalianFiltering24(char* input, char* output) woiddianFordalianFiltering24(char* input, char* output) woiddianFordalianFiltering24(char* input, char* output, double degree)  woiddianFiltering24(char* input, char* output, input, char* output, double degree)  woiddianFiltering24(char* input, char* output, double degree)  woiddianFiltering24(char* input, char* output, double degree)  woid with Xi input 是输入文件名,output 是输出文件名。 degree 是旋转的度数。 支持 24 位 BMP 图像。  woiddianFiltering24(char* input, char* output, double degree)  woid with Xi input 是输入文件名,output 是输出文件名。 degree 是旋转的度数。 支持 24 位 BMP 图像。  woiddianFiltering24(char* input, char* output, double degree)  woid with Xi input 是输入文件名,output 是输出文件名。 degree 是旋转的度数。 支持 24 位 BMP 图像。  woid
void MeanFiltering24(char* input, char* output)
input, char* output) void MedianFiltering24(char* input, char* output) void ZoomOutAndZoomIn(char* input, char* output, double value) void Translation24(char* input, char* output, int x, int input, char* output, int x, int y)  void Mirror24(char* input, char* output) void Rotate24(char* input, char* output) void Rotate24(char* input, char* output, double degree) void SivenThresholdMethod(char* 是输入文件名,output 是输出文件名。 Employed BMP 图像。 char* output 是输入文件名,output 是输出文件名。 degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。 char* output, double degree 是旋转的度数。支持 24 位 BMP 图像。
void MedianFiltering24(char* input, char* output) void ZoomOutAndZoomIn(char* 缩放(双线性插值),input 是输入文件名,output 是输入文件名,output 是输出文件名。value 是放大倍数,如 value=0.5。支持 24 位 BMP 图像。 void Translation24(char* input, char* output, int x, int y) void Mirror24(char* input, char* output) void Rotate24(char* input, char* output) void Rotate24(char* input, char* output, double degree) Void SivenThresholdMethod(char* Embry 24 char* input, char* output, double degree, BMP 图像。 Void SivenThresholdMethod(char* Embry 24 char* input, char* output, double degree, BMP 图像。 Void SivenThresholdMethod(char* Embry 24 char* input, char* output, double degree, BMP 图像。 Void SivenThresholdMethod(char* Embry 24 char* input, char* output, double degree, BMP 图像。 Void SivenThresholdMethod(char* Embry 24 char* input, char* output, double degree, BMP 图像。 Void SivenThresholdMethod(char* Embry 24 char* input, char* output, double degree, BMP 图像。
input, char* output) 输出文件名。支持 24 位 BMP 图像。 void ZoomOutAndZoomIn(char* input, char* output, double value)
woid ZoomOutAndZoomIn(char* input, char* output, double value) woid Translation24(char* input, char* output, int x, int y) woid Mirror24(char* input, char* output) woid Rotate24(char* input, char* output) woid Rotate24(char* input, char* output, double degree) woid SivenThresholdMethod(char* input, char* output, double degree, woid size input, double degree degree degree degree,
walue) 如 value=0.5。支持 24 位 BMP 图像。  void Translation24(char* input, char* output, int x, int y)
woid Translation24(char* input, char* output, int x, int y)
input, char* output, int x, int y)  void Mirror24(char* 镜像, input 是输入文件名, output 是输出input, char* output)  void Rotate24(char* input, char* output, double degree)  void Rotate24(char* input, char* output, double degree)  woid GivenThresholdMethod(char* 文件名, output 是输出文件名。 支持 24 位 BMP 图像。  void Rotate24(char* input 是输入文件名, output 是输出文件名。 degree 是旋转的度数。 支持 24 位 BMP 图像。  woid 给定阈值法处理图像,使图片黑白化, input 是输入文件名, output 是输出文件名。
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 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 文件名。degree 是旋转的度数。支持 24 位 BMP 图像。  void egree) BMP 图像。  void 给定阈值法处理图像,使图片黑白化,input 是输入文件名,output 是输出文件名。
像。 void Mirror24(char* 镜像, input 是输入文件名, output 是输出 文件名。支持 24 位 BMP 图像。 void Rotate24(char* 旋转, input 是输入文件名, output 是输出 文件名。degree 是旋转的度数。支持 24 位 BMP 图像。 void GivenThresholdMethod(char* 是输入文件名, output 是输出文件名。
voidMirror24(char* input, char* output)镜像, input 是输入文件名, output 是输出 文件名。支持 24 位 BMP 图像。voidRotate24(char* input, char* output, double degree)旋转, input 是输入文件名, output 是输出 文件名。degree 是旋转的度数。支持 24 位 BMP 图像。void给定阈值法处理图像,使图片黑白化, input 是输入文件名, output 是输出文件名。
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 degree) 发件名。degree 是旋转的度数。支持 24 位 BMP 图像。 void 给定阈值法处理图像,使图片黑白化, input 是输入文件名, output 是输出文件名。
voidRotate24(char* input, char* degree)旋转, input 是输入文件名, output 是输出 文件名。degree 是旋转的度数。支持 24 位 BMP 图像。void GivenThresholdMethod(char*给定阈值法处理图像,使图片黑白化, input 是输入文件名, output 是输出文件名。
input, char*output, double degree)文件名。degree 是旋转的度数。支持 24 位 BMP 图像。void GivenThresholdMethod(char*给定阈值法处理图像,使图片黑白化,input 是输入文件名,output 是输出文件名。
degree)BMP 图像。void给定阈值法处理图像,使图片黑白化,inputGivenThresholdMethod(char*是输入文件名,output 是输出文件名。
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。
GivenThresholdMethod(char* 是输入文件名, output 是输出文件名。
input, char* output, int threshold 是给定的阈值,如
threshold threshold=100。支持 24 位 BMP 图像。
void   迭代阈值法处理图像,使图片黑白化,input
IterativeThresholdMethod(cha
r* input, char* output) 24 位 BMP 图像。 void 0stu (大津法) 阈值分割, input 是输入文
Ostu (人样法) 國值分割,Input 定制人义 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
图像。

	<b>マカル・・・ロか</b> ン <b>カ</b> り <b>カ</b>
void CorrectMethod(char*	正确法, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 24 位 BMP 图像。
void	对图像分理出其中的 RGB 分量并分别保存
ChannelSeparation1(char*	为独立的图像, input 是输入文件名,
input, char* Routput, char*	Routput 是红色通道图像,Goutput 是绿色
Goutput, char* Boutput)	通道图像,Boutput 是绿色通道图像。支持
	24 位 BMP 图像。
void ReverseColor(char*	对灰度图进行反色, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 8 位 BMP 图像。
Image1* LoadImage1(char*	BMP 图像读取, input 是输入文件名。支持
input)	8位和24位BMP图像。
	返回 Imagel 型数据,Imagel 型数据的结构
	如下:
	typedef struct
	{
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
	} Image1;
void SaveImage1(char*	将 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
output, imagei* img/	
	图像数据。支持 8 位和 24 位 BMP 图像。
	Imagel 型数据的结构如下:
	typedef struct
	{
	int width;
	int height;
	int channels; //图像通道数
	unsigned char* Data; //像素数据
	}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 $a=(255.0-m*(g2-g1))/(255.0-$
α/	(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)	output Zini II Zin o Zini o E biii o Kini
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)	加出入口石。 关系 0 图 BM 图像。
void	腐蚀操作, input 是输入文件名, output 是
CorrosionOperation(char*	输出文件名。支持8位BMP图像。
input, char* output)	
void Operation1(char*	开操作, input 是输入文件名, output 是输
input, char* output)	出文件名。支持8位BMP图像。
void Closed1(char*	闭操作, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void Negativel(char*	图像反色, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 24 位 BMP 图像。
void Negative(char*	图像反色, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void ImageSynthesis(char*	图像合成。
input1, char* input2, char*	
output)	
void BlackWhite(char*	黑白化,支持8位和24位BMP图像。T是
input, char* output, float	阈值,border 是边界范围,如: T=50,
T, int border)	border=0。
void Mosaic(char*	马赛克化图像, w 和 h 是输出图像的宽和
input, char* output, int w, int	高。支持 PNG 图像。
h)	
void	缩放图片(最近邻插值法)。
TransformShapeNearest(char*	
inputFile, char* output,	
unsigned int newWidth,	
unsigned int newHeight)	As Martin Land As In L
void	缩放图片(双线性插值法)。
TransformShapeLinear(char*	
inputFile, char* output,	
unsigned int newWidth,	
unsigned int newHeight)	<b>网</b> 换的红音角度的埃拉
Void TransformShanoWhirl (abort	图像的任意角度的旋转。
TransformShapeWhirl(char* inputFile, char* output,	
float angle)	
void	图像的镜像翻转。
TransformShapeUpturn(char*	
inputFile, char* output, int	
inputific, char output, the	

a)	
void	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)	<b>□</b> • /
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	— 值 图 (
ar* inputFile, char* output,	代, area312e=23 次日旦/
int areaSize)	
· ·	反色。
void TransformColorOpposite(chark	(X巴。
TransformColorOpposite(char*	
inputFile, char* output)	古宝园均海儿 (八生计符 为用更加矛和)
void	直方图均衡化(分步计算,效果更加柔和)。
TransformColorHistogramPart( char* inputFile.char*	
output) void	   直方图均衡化(整体计算,效果更加尖锐)。
	旦刀图均偶化(釜徑口昇,双米史加大號)。  
TransformColorHistogramAll(c	
har* inputFile, char* output)	<b>半</b> 和操作(白皂ツ)
void KernelsUseDIY(char*	卷积操作(自定义)。 
<pre>inputFile, char* output, double* kernels, int</pre>	
,	
areaSize, double modulus)	
Void WayafiltaringMadian (abary	中值滤波。
WavefilteringMedian(char*	
inputFile, char* output) void	
Void   WavefilteringGauss(char*	高斯滤波。a=9,b=1.0/16。 高斯滤波卷积核:
inputFile, char*	尚别認仅仓保核: double KERNELS_Wave_Gauss[9] =
output, double*	
KERNELS_Wave_Gauss, int	1, 2, 1,
a, double b)	
a, doubte b)	2, 4, 2, 1, 2, 1
	};

```
void
                             低通滤波。areaSize=9, modulus=1。
                             // 低通滤波卷积核 LP1
Wavefiltering LowPass(char*
                             double KERNELS Wave LowPass LP1[9] =
inputFile,
            char*
                     output,
double*
                 kernels, int
                                 1 / 9.0, 1 / 9.0, 1 / 9.0,
areaSize, double modulus)
                                 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
                             高通滤波。areaSize=9, modulus=1。
void
                             // 高通滤波卷积核 HP1
WavefilteringHighPass(char*
                             double KERNELS Wave HighPass HP1[9] =
inputFile,
            char*
                     output,
double*
                 kernels, int
areaSize, double modulus)
                                 -1, -1, -1,
                                 -1, 9, -1,
                                 -1, -1, -1
                             };
                             // 高通滤波卷积核 HP2
                             double KERNELS_Wave_HighPass_HP2[9] =
                                 0, -1, 0,
                                -1, 5, -1,
                                 0, -1, 0
                             };
                             // 高通滤波卷积核 HP3
                             double KERNELS_Wave_HighPass_HP3[9] =
```

```
1, -2, 1,
                                -2, 5, -2,
                                1, -2,1
void
                             均值滤波。areaSize=25, modulus=1.0 /
Wavefiltering Average(char*
                             25。
                             // 均值滤波卷积核
inputFile, char*
output, double*
                             double KERNELS_Wave_Average[25] =
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, 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,
                             doub1e
double*
                             KERNELS_Edge_difference_vertical[9] =
                kernels, int
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
```

```
// Sobel X边缘检测卷积核
KernelsUseEdgeSobel(char*
inputFile,
            char*
                             double KERNELS Edge Sobel X[9] =
                    output,
double*
         kernels1,
                    double*
kernels2)
                                 -1, 0, 1,
                                -2, 0, 2,
                                 -1, 0, 1
                             };
                             // Sobel Y边缘检测卷积核
                             double KERNELS_Edge_Sobel_Y[9] =
                                -1, -2, -1,
                                 0, 0, 0,
                                 1, 2, 1
                             };
                             Laplace 边缘检测。 areaSize=9,
void
EdgeDetectionLaplace(char*
                             modulus=1.
inputFile, char*
                             // Laplace 边缘检测卷积核 LAP1
                             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] =
```

	T
	1, -2, 1,
	-2, 8, -2,
	1, -2, 1
woid MannhalagyEnagian (shany	腐蚀。
void MorphologyErosion(char*	
inputFile, char*	// 腐蚀卷积核   double
output, double* kernels)	KERNELS Morphology Erosion cross[9] =
	KERNELS_MOTPHOTOGY_ETOSTOH_CTOSS[9] =
	0, 1, 0,
	1, 1, 1,
	0, 1, 0
	};
void	膨胀。
MorphologyDilation(char*	// 膨胀卷积核
inputFile, char*	double
output, double* kernels)	KERNELS Morphology Dilation cross[9]
	=
	{
	0, 1, 0,
	1, 1, 1,
	0, 1, 0
	};
void FaceDetection(char*	人脸检测。Threshold 是二值化时的阈值,
input, char* output, double*	areaSize1=25, modulus1=1.0/25, 对于
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
areaSize, unsigned char	法,5表示绿色分量法,6表示蓝色分量法;
Threshold, unsigned char	GrayscaleMode=1, areaSize=25, B=0, G=0,
B, unsigned char G, unsigned	R=200, a=1 表示选择均值滤波预处理图像,
char R, int a, int b, int	a=2 表示选择高通滤波预处理图像,a=3 表
areaSizel, double modulus1)	示选择低通滤波预处理图像, a=4 表示选择
	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, inFileName 是加密图像文件名,
inFileName, char*	outFileName 是解密图像文件名,key 是密
outFileName, char key)	钥,如 key=255。支持 8 位、24 位和 32 位

	BMP 图像。
void Compress8(string	图像压缩, input 是输入文件名, output 是
input, string output)	输出文件名。支持 8 位 BMP 图像。
void Decompression(string	图像解压, input 是输入文件名, output 是
input, string output)	输出文件名。支持 8 位 BMP 图像压缩后的结
	果文件。
void HorizontalMirror(char*	水平镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void MirrorVertically(char*	垂直镜像, input 是输入文件名, output 是
input, char* output)	输出文件名。支持 8 位 BMP 图像。
void XMirroring(char*	X 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void YMirroring(char*	Y 镜像, input 是输入文件名, output 是输
input, char* output)	出文件名。支持 8 位 BMP 图像。
void ImageConvolution(char*	图像卷积, input 是输入文件名, output 是
input, char* output, double**	输出文件名。Kernel 是卷积核,如 double
Kernel, int n, int m)	Kernel[3][3] = {{-0.225, -0.225-
	0. 225}, {-0. 225, 1, -0. 225}, {-0. 225, -
	0.225,-0.225}}; n 是 Kernel 的第一维的
	大小, m 是 Kernel 的第二维的大小, 形如
	Kernel[n][m]。支持24位BMP图像。
void SpatialMeanFiter(char*	空间均值过滤器,参考: radius=3。
input, char* output, int	
radius)	
void	空间中值过滤器,参考: radius=3。
SpatialMedianFiter(char*	
input, char* output, int	
radius)	
void SpatialMaxFiter(char*	空间最大过滤器,参考: radius=3。
input, char* output, int	
radius)	
void SpatialMinFiter(char*	空间最小过滤器,参考: radius=3。
input, char* output, int	
radius)	
void SpatialGaussFiter(char*	空间高斯过滤器,参考: radius=3。
input, char* output, int	
radius)	
void	空间统计滤波器,参考: radius=3, T=0.2。
SpatialStatisticalFiter(char	
* input, char* output, int	
radius, float T)	
void FFTAmp(char*	FFT 放大器,参考:inv=false。
input, char* output, bool inv)	
void FFTPhase(char*	FFT 相位,参考:inv=false。

input, char* output, bool inv)	
void STDFT1(char*	参考: inv=false。
input, char* output, bool inv)	
void STDFT2(char*	参考: inv=false。
input, char* output, bool inv)	
void SpectrumShaping(char*	图像频域滤波, FFT 变换相位谱,
input, char* inputMsk, char*	inputMsk 是输入的掩膜图像名。
output)	
void Translation(char*	图像平移, input 是输入文件名, output 是
input, char* output, int x, int	输出文件名。x和y是在X轴和Y轴平移的
y, unsigned char color)	量,以右为正向,color 是平移后非原图区
	域填充的颜色,如 color=100。支持 8 位 BMP
	图像。
void 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)	具汇机括估计土坝构 : 目於入文件
void   Image Scaling Near of the inches	最近邻插值法去栅格, input 是输入文件 名, output 是输出文件名。1x 和 1y 是长和
ImageScalingNearestNeighborI	名,output 是制出义件名。1x 和 1y 是长和   宽需要缩放的倍数。支持 8 位 BMP 图像。
nterpolation(char*	见而女细从印旧数。 又付 0 世 DMF 图像。
input, char* output, float 1x, float 1y)	
void	   双线性插值法去栅格, input 是输入文件
ImageScalingBilinearInterpol	名, output 是输出文件名。1x 和 1y 是长和
ation(char* input, char*	宽需要缩放的倍数。支持8位BMP图像。
ation (chair input, chair	见而安组从时旧效。又打 0 型 DMF 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)	从旧数(元四万一数7。 支持 四面 国家。
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)	从旧数(元川小数)。 文刊 四皿 国家。
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
input, int with the height)	顺序一一对应。
BMPMat**	图像复原, width 是原图像的宽, height 是
ImageRestoration(vector < BMPM	原图像的高。
at>	
<pre>color, vector<vector<coor> &gt;</vector<coor></pre>	
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)	
void ImageRotation(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 8 位 BMP 图像。angle 是
angle)	要旋转的角度。
void Rotation8(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 8 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标, color 是旋转后非原
color)	图区域的填充颜色。
void Rotation24(char*	图像旋转, input 是输入文件名, output 是
input, char* output, double	输出文件名。支持 24 位 BMP 图像。Angle 是
Angle, int x1, int y1, int	要旋转的角度数; x1、y1、x2、y2 是旋转所
x2, int y2, unsigned char	围绕的中心点的坐标; red、green、blue 分
red, unsigned char	别是旋转后非原图区域要填充的颜色的红

groon ungigned shor blue	绿蓝分量。
green, unsigned char blue)	· — · · —
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。
input, char* output, double	
a, double b)	
void NoiseGauss(char*	高斯噪声,参考: mean=0, delta=31。
input, char* output, float	
mean, float delta)	
void NoiseRayleigh(char*	瑞利噪声,参考: a=0, b=200。
input, char* output, float	
a, float b)	
void NoiseExp(char*	指数噪声,参考: a=0.1。
input, char* output, float a)	
void NoiseImpulse(char*	椒盐噪声,参考: a=0.2, b=0.2。
input, char* output, float	
a, float b)	
void grayToColor(FILE*	灰色转伪彩色, input 是输入文件, output
input, FILE* output)	是输出文件。支持 8 位和 24 位 BMP 图像。
void ImageThinning(char*	图像细化, input 是输入文件名, output 是
input, char* output, char**	输出文件名。支持 4 位 BMP 图像。n 是 str
str, int n, int 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)	
int	返回图像像素的最大值,filename 是输入
MaximumValueOfImagePixels(ch	的图像文件名。支持 8 位和 24 位 BMP 图像。

ar* filename)	
float	  返回图像像素的均值,filename 是输入的
AverageValueOfImagePixels(ch	图像文件名。支持 8 位和 24 位 BMP 图像。
ar* filename)	
double	返回图像像素的标准差,filename 是输入
StandardDeviationOfImagePixe	的图像文件名。支持 8 位和 24 位 BMP 图像。
ls (char* filename)	门国家人口有6人们 6 医月 21 医 Bin 国家。
double EntropyOfImage(char*	返回图像的熵,支持8位和24位BMP图像。
filename)	应自国际们间, 文内 0 四/H 21 匝 Dim 国家。
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
ar Trename)	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)	Interpolation o Ainterpolation 16
void HSV(char* input, char*	图像色调饱和度明度调节,参考: h=120,
output, int h, int s, int v)	s=60, v=20.
void ColorTransfer1(char*	颜色转移,支持 BMP 图像。
input1, char* input2, char*	
output)	
void OilpaintFilter(char*	油画滤镜,参考: radius=10, smooth=100。
input, char* output, int	和画版成,多·3. IddId5 10, 5mooth 1008
radius, int smooth)	
void HaloFilter(char*	晕角滤镜,参考: ratio=100。
input, char* output, int	
ratio)	
void GrayHistogram(char*	灰度直方图,参考: hWidth=256,
input, char* output, int	hHeight=100.
hWidth, int hHeight)	J
void RedHistogram(char*	红色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100.
hWidth, int hHeight)	
void GreenHistogram(char*	绿色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100.
hWidth, int hHeight)	
void BlueHistogram(char*	蓝色通道直方图,参考: hWidth=256,
input, char* output, int	hHeight=100。
hWidth, int hHeight)	-
void	直方图均衡化, input 是输入文件名,
HistogramEqualization2(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output, int	BMP 图像。imgBit 是输入图像的位数。
imgBit)	

void	直方图均衡化, input 是输入文件名,
HistogramEqualization3(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output)	BMP图像。
void	直方图均衡化, input 是输入文件名,
HistogramEqualization4(char*	output 是输出文件名。支持 8 位和 24 位
input, char* output)	BMP 图像。input 是输入文件名称,out 是
	输出文件名称。
void	直方图均衡化,参考: hWidth=256,
HistogramEqualization(char*	hHeight=100。
input, char* output, int	
hWidth, int hHeight)	
void	灰度直方图,参考: hWidth=256,
GrayHistogramEqualization(ch	hHeight=100。
ar* input, char* output, int	
hWidth, int hHeight)	
void	红色通道直方图,参考: hWidth=256,
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	intergrit-100°
hWidth, int hHeight)	蓝色运送古之图 乡类 1W: 141-056
void	蓝色通道直方图,参考: hWidth=256,
BlueHistogramEqualization(ch	hHeight=100。
ar* input, char* output, int	
hWidth, int hHeight)	+ F / D / 4 1 1 1 1 0 5 0
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。
<u> </u>	J

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 Threshold=0.001。
Threshold)	
void OTSUProcessing(char*	大津法处理, input 是输入文件名, output
input, char* output)	是输出文件名。支持8位BMP图像。
void OBJtoTGA(char*	OBJ 转 TGA。
input, char* output, int	
width, int height)	
void ToRIM(char* input, char*	一般图像转到 RIM 图像,支持 PNG、JPG 和
output)	TGA 图像。
void ToImage(char*	RIM 图像转到一般图像,支持 PNG、JPG 和
input, char* output, int	TGA 图像。jpg_quality=25。
jpg_quality)	Ton Ed By 198_quarrey 200
void	将 1 位深度的单色 BMP 图片转成热敏打印
ImprimanteThermique(char*	机的位图打印输出,支持的热敏打印机的位
input, char* output, ARRAY3	图打印指令为〈strong〉ESC *〈/strong〉指
skip cmd, unsigned short	令。
PRINTER TYPE BMP, unsigned	く。 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,
a, unsigned char b)	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,
woid Whit-D-1/-	FILE_TYPE_AD=(0x4441), a=0x80, b=1.
void WhiteBalance(const	白平衡。
char* input, const char*	
output)	Cohol 質 乙
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)	Conny 質 Z
void Canny (char* input, char*	Canny 算子, magnScale=0.35,
output, double	lowThreshold=55, highThreshold=120。支 共 DCM 和 DDM 図典
magnScale, double	持 PGM 和 PBM 图像。
lowThreshold, double	
highThreshold)	図台/k +breeshelld=100 111-0
void BlackWhite(char*	黑白化,threshold=100,background=0。

	Lett pay to pay Ella
input, char* output, int	支持 PGM 和 PBM 图像。
threshold, int background)	
void	区域连通,threshold=100,background=0,
ConnectedComponents(char*	threshold1=100。支持 PGM 和 PBM 图像。
input, char* output, int	
threshold, int background, int	
threshold1)	
void CleanImage(char*	清洁图像。支持 PGM 和 PBM 图像。
input, char* output)	
void NoiseImage(char*	噪声化图像,probability=0.1。支持 PGM
input, char* output, float	和 PBM 图像。
probability)	
void	圆检测。scale1=1,gamma1=1.0,
HoughTransformCircle1(char*	magnScale=0.5 , lowThreshold=85 ,
input, char* output, double	highThreshold=150, scale=0, gamma=1.0,
sigma, int kernelSize, int	sigma 和 kernelSize 用于平滑的高斯 5x5
scale, double gamma, double	内核, sigma=1.0, kernelSize=5。
magnScale, double	如果 scale==0,则值保持不变,但如果它们
lowThreshold, double	低于0或高于255,则它们分别设置为0或
highThreshold, int scalel,	255。
double gammal)	如果 scale! =0,则对值进行缩放,使得最
double gamma1)	小值为零,最大值为 255。
	设置 gamma 值允许指数缩放,启用
	以且 gamma 国九川百数相从, 石円 gamma==1.0。
	支持 PGM 和 PBM 图像。
void	圆检测。scale1=1,gamma1=1.0,
HoughTransformCircle2(char*	magnScale=0.5 , lowThreshold=85 ,
input, char* output, int	highThreshold=150, scale=0, gamma=1.0,
number, int minDist, double	sigma 和 kernelSize 用于平滑的高斯 5x5
sigma, int kernelSize, int	内核,sigma=1.0,kernelSize=5,number=10
scale, double gamma, double	表示图像的目视检查有 10 个圆圈,
magnScale, double	minDist=35。
lowThreshold, double	如果 scale==0,则值保持不变,但如果它们
highThreshold, int scalel,	低于0或高于255,则它们分别设置为0或
double gammal)	255。
	如果 scale! =0,则对值进行缩放,使得最
	小值为零,最大值为 255。
	设置 gamma 值允许指数缩放,启用
	gamma==1.0。
	支持 PGM 和 PBM 图像。
double**	圆检测。scale1=1,gamma1=1.0,
HoughTransformCircle3(char*	magnScale=0.5 , lowThreshold=85 ,
input, char* output, int	highThreshold=150, scale=0, gamma=1.0,
number, int minDist, double	sigma 和 kernelSize 用于平滑的高斯 5x5

sigma, int kernelSize, int	内核,sigma=1.0,kernelSize=5,number=10
scale, double gamma, double	表示图像的目视检查有 10 个圆圈,
magnScale, double	minDist=35。
lowThreshold, double	如果 scale==0,则值保持不变,但如果它们
highThreshold, int scale1,	低于0或高于255,则它们分别设置为0或
double gammal)	255。
_	如果 scale! =0,则对值进行缩放,使得最
	小值为零,最大值为 255。
	设置 gamma 值允许指数缩放,启用
	gamma==1.0°
	返回以(vCenter,hCenter)和半径
	(vRadius, hRadius) 为中心的椭圆数据,
	共有 number 组数据,每组包含一个椭圆数
	据,第一个元素是 vCenter,第二个元素是
	hCenter,第三个元素是 vRadius,第四个元
	素是 hRadius。
	支持 PGM 和 PBM 图像。
void	形状边缘检测, CANNY THRESH4=35,
ShapeEdgeDetection1(char*	CANNY_blur4=7。支持 PNG 图像。
input, char* output, unsigned	CVIVIT_DIGIT4-1。 文刊 ING 图像。
char CANNY_THRESH4, int CANNY blur4)	
void	形状边缘检测,CANNY_THRESH4=35,
	CANNY blur4=7。支持 PNG 图像。
ShapeEdgeDetection2(char* input, char* output, unsigned	CAMMI_DIGIT4-1。 文书 FING 国家。
1 CANDILL WILD DOLLAR	
char CANNY_THRESH4, int CANNY blur4)	
void	
ShapeEdgeDetection3(char*	CANNY BLUR=12。支持 PNG 图像。
	CANNI_DLUK-12。 文书 LNG 国家。
input, char* output, unsigned	
char CANNY_THRESH, int	
CANNY_BLUR) void	形状边缘检测, CANNY_THRESH=50,
	—
ShapeEdgeDetection4(char*	CANNY_BLUR=12。支持 PNG 图像。
input, char* output, unsigned char CANNY THRESH, int	
CANNY_BLUR) void	形状边缘检测,CANNY_THRESH2=10,
ShapeEdgeDetection5(char*	IN AC
	CUMMI_DEGIZ=2。 文建 LING 思像。
input, char* output, unsigned	
char CANNY_THRESH2, int	
CANNY_BLUR2)	II
void Shana Edga Data ation 6 (ahany	形状边缘检测,CANNY_THRESH2=10,
ShapeEdgeDetection6(char*	CANNY_BLUR2=2。支持 PNG 图像。

input, char* output, unsigned char CANNY_THRESH2, int	
CANNY_BLUR2)	TO JD ) L. Mr. IA ) Ed
void	形状边缘检测,CANNY_THRESH3=45,
ShapeEdgeDetection7(char*	CANNY_blur3=10。支持 PNG 图像。
input, char* output, unsigned	
char CANNY_THRESH3, int	
CANNY_blur3)	
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

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。
<pre>void Decode(char* input, char*</pre>	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。
<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。
uint8_t* AESdecrypt(uint8_t*	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 FileCompress(char *input ,	文件压缩, input 是输入文件名,
char *output)	output 是输出文件名。
void FileDecompression(char	文件压缩结果解压缩, input 是输入
*input , char *output)	文件名,output 是输出文件名。

## 高级算子

void BlobAnalysis(char*	Blob 分析, c1 和 c2 是与颜色相关的
input, char* output, int c1, int c2)	参数,参考: c1=128, c2=127。支持 BMP
	图像。
void BlobAnalysis1(char*	Blob 分析, c1 和 c2 是与颜色相关的
input, char* output, int c1, int c2)	参数,参考: c1=128, c2=127。支持 BMP
	图像。
void	验 证 码 生 成 。
VerificationCodeGeneration(char*	sigma=10, noise_type=2, a=10, b1=12
inputText, char* output, int	8, b2=127, b3=2, b4=8, b5=12, b6=0,
num, int foint, int a, int b1, int	b7=1, b8=-100, b9=-100, b10=1,
b2, int b3, int b4, int b5, int b6, int	b11=3 , b12=6 , int b13=40 ,
b7, int b8, int b9, int b10, int	foint=30, num 是字符数
b11, int b12, int b13, double	量,depth=1,spectrum=3,shared=0。

```
sigma, unsigned int noise_type, int
width, int
           height, int
                       depth, int
spectrum, bool shared)
void
           CornerDetection(char*
                                  角点检测, threshold=10000, k=0.06,
input, char*
                    output, float
                                  sigma=1.0, width=640, height=480,
threshold,
             float
                     k,
                           float
                                  channels=1。支持 PNM 图像。
sigma, int width, int height, int
channels)
                                  角点检测,返回角点数据。
vector < Keypoint >
                                  threshold=10000
CornerDetection1(char*
                                                       k=0.06
                                  sigma=1.0, width=640, height=480,
input, char*
                    output, float
threshold,
             float
                                  channels=1。支持 PNM 图像。
                     k,
                           float
sigma, int width, int height, int
                                  需引入以下结构体
                                  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。
channels, float
                                  需引入以下命名空间:
              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
output, float sigma)
                                  持多种图像格式。
void Cornerness(char* input, char*
                                  角点检测,参考: sigma=2, method=0。
                                  支持多种图像格式。
output, float sigma, int method)
     Corners(char*
                                  角点检测, 参考: sigma=2,
void
                     input, char*
                                  thresh=0.4, window=5, nms=3,
output, float sigma, float thresh,
int
      window.
                int
                      nms,
                             int
                                  corner method=0。支持多种图像格
corner method)
                                  式。
void FindMatch(char* input1, char*
                                  特征匹配,参考: thresh3=5, k=10000,
input2, char*
                                  cutoff=50, thresh4=5, sigma=2,
                    output, float
thresh3, int k, int cutoff, float
                                  thresh=0.4, window=5, nms=3,
thresh4, float
                 sigma,
                           float
                                  corner method=0 , sigma1=2 ,
thresh, int window, int nms, int
                                  thresh1=0.4, window1=5, nms1=3,
corner method, float sigmal, float
                                  corner method1=0 , sigma2=2 ,
thresh1, int window1,
                      int nms1,
                                  thresh2=0.4, window2=5, nms2=3,
```

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

corner\_method2=0 , sigma5=2 , thresh5=0.4, window5=7, nms5=3, corner\_method5=0 , sigma6=2 , corner\_method6=0 , thresh6=0.3 , window6=7 , nms6=3 , inlier\_thresh6=5 , iters6=1000 , cutoff6=50 , acoeff6=0.5 。 支持多种 图像格式。

vector (Descriptor)

HarrisCorner(char\* input, char\*
output, float sigmal, float
threshl, int windowl, int nmsl,
int corner methodl)

角点检测,返回检测结果。参考: sigma1=2,thresh1=0.4,window1=5, nms1=3,corner\_method1=0。支持多种 图像格式。

需引入以下结构体:

```
struct Point {
  double x, y;
```

```
Point() : x(0), y(0) {}
Point(double x, double y) :
x(x), y(y) {}
};
struct Descriptor {
Point p;
vector<float> data;

Descriptor() {}
Descriptor(const Point& p) :
p(p) {}
```

vector < Match >

MatchDescriptors (char\* input1, char\* input2, char\* output, float sigma1, float thresh1, int window1, int nms1, int corner\_method1, float sigma2, float thresh2, int window2, int nms2, int corner\_method2)

描述匹配项,返回描述结果。参考: sigma1=2, thresh1=0.4, window1=5, nms1=3 , corner\_method1=0 , sigma2=2, thresh2=0.4, window2=5, nms2=3, corner\_method2=0。支持多种图像格式。

```
需引入以下结构体:
```

```
struct Point {
  double x, y;
```

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

```
struct Descriptor {
  Point p;
  vector<float> data;
 Descriptor() {}
 Descriptor(const Point& p) :
p(p) {}
};
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) {}
         operator < (const
  bool
                            Match&
other)
                            return
distance other. distance; }
```

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 corner method1, float nms1, float thresh2. sigma2, int window2, int nms2, 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)

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。

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, float thresh, int window, int nms, int corner method, float sigmal, float thresh1, int window1, nms1, int corner method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, int nms5, int corner method5, float sigma6, int corner\_method6, float thresh6, int window6, int nms6, float inlier\_thresh6, int iters6, int cutoff6, float acoeff6)

thresh1=0.4, window1=5, nms1=3, corner\_method1=0 , sigma2=2 , thresh2=0.4, window2=5, nms2=3, corner\_method2=0 , sigma5=2 , thresh5=0.4, window5=7, nms5=3, corner\_method5=0 , sigma6=2 , corner\_method6=0 , thresh6=0.3 , window6=7 , nms6=3 , inlier\_thresh6=5 , iters6=1000 , cutoff6=50 , acoeff6=0.5 。 支持多种图像格式。

void Cylindrical (char\* input1, char\* input2, char\* output, float f1, float f2, float thresh3, int k, int cutoff, float thresh4, float sigma, float thresh, int window, int nms, int corner method, float sigmal, float thresh1, int window1, int nms1, int corner method1, float sigma2, float thresh2, int window2, int nms2, int corner method2, float sigma5, float thresh5, int window5, int nms5, int corner method5, float sigma6, int corner method6, float thresh6, int window6, int nms6, float inlier thresh6, int iters6, int cutoff6, float acoeff6, float sigma7, int corner method7, float thresh7, int window7, int nms7, float inlier\_thresh7, int iters7, int cutoff7, float acoeff7)

角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3 , corner method1=0 sigma2=2, thresh2=0.4, window2=5, nms2=3 , corner method2=0 sigma5=2, thresh5=0.4, window5=7, nms5=3, corner method5=0 sigma6=2, corner\_method6=0 , thresh6=0.3, window6=7, nms6=3, inlier\_thresh6=5, iters6=1000, cutoff6=50, acoeff6=0.5, sigma7=2, corner method7=0, thresh7=0.3, window7=7 nms7=3inlier thresh7=5, iters7=1000, cutoff7=50, acoeff7=0.5。支持多种 图像格式。

void 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,

角点检测。参考: f1=500, f2=500, thresh3=5, k=10000, cutoff=50, thresh4=5, sigma=2, thresh=0.4, window=5, nms=3, corner\_method=0, sigma1=2, thresh1=0.4, window1=5, nms1=3, corner\_method1=0, sigma2=2, thresh2=0.4, window2=5, nms2=3, corner\_method2=0, sigma5=2, thresh5=0.4, window5=7,

```
float thresh2, int window2,
                                  nms5=3,
                                             corner method5=0
                             int
            corner method2, float
                                  sigma6=2, corner method6=0,
nms2.
       int
                                  thresh6=0.3, window6=7, nms6=3,
sigma5,
          float
                  thresh5,
                             int
window5,
            int
                   nms5.
                             int
                                  inlier thresh6=5, iters6=1000,
corner method5, float sigma6,
                                  cutoff6=50, acoeff6=0.5, sigma7=2,
                            int
corner method6,
                                  corner method7=0, thresh7=0.3,
                 float
                        thresh6,
                                  window7=7
int window6,
               int
                    nms6,
                           float
                                                     nms7=3
inlier thresh6, int iters6,
                                  inlier thresh7=5, iters7=1000,
                             int
cutoff6.
           float
                   acoeff6, float
                                  cutoff7=50, acoeff7=0.5。支持多种
sigma7, int corner method7, float
                                  图像格式。
thresh7, int window7, int nms7,
float inlier_thresh7, int iters7,
int cutoff7, float acoeff7)
     FindLinel(char*
                                  直线检测,返回直线的 theta 和 rho,
                       input, int
                                  支持 RAW 图像。
width, int height)
                    hough param*
                                  直线检测,返回检测结果,支持 RAW 图
struct
FindLine2(char* input, int width,
int height)
                                  需引入以下结构体:
                                  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;
vector<struct
                   hough param*>
                                  直线检测,返回检测结果数组,支持
FindLine3(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;
int* FindCircle1(char* input, int
                                 圆检测,返回圆心的坐标和圆的半径,
width, int height, float sigma, int
                                 支持 RAW 图像。sigma=1.4,tmin=70,
tmin, int tmax)
                                 tmax=150.
                                 圆检测,返回检测结果,支持 RAW 图
struct
             hough param circle*
FindCircle2(char*
                                 像。sigma=1.4, tmin=70, tmax=150。
                   input,
                             int
width, int height,
                   float sigma,
                                 需引入以下结构体:
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,返回检测结果数组,支持 RAW
vector<struct
hough_param_circle*>
                                 图像。sigma=1.4,tmin=70,tmax=150。
                                 需引入以下结构体:
FindCircle3(char*
                   input,
                             int
                                 struct point {
width, int height,
                   float sigma,
int tmin, int tmax)
                                     int x;
                                     int y;
                                 };
                                 struct hough param circle {
                                     int a;
                                     int b;
                                     int radius;
                                     int resolution;
                                     int thresh:
                                     struct point *points;
                                     int points size;
                                 模板匹配, a=0.5, b=0, a1=0.5,
                       Template*
struct
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
                       al, double | struct Template
            a, double
bl, double c)
                                      double corelatia;
                                      unsigned int nrFereastra;
                                      unsigned int cifra;
                                      unsigned int ly1;
                                      unsigned int rly;
                                      unsigned int 1x1;
                                      unsigned int rlx;
vector<SearchResult>
                                  模板匹配,返回以目标位置的左上角
TemplateMatching(char*
                                  坐标和相似值组成的数组。
                                  支持 PNG 图像。
input, char* Template)
                                  需引入以下结构体:
                                  struct SearchResult {
                                      int x, y;
                                      double value;
                                  图像标定。
void
          ImageCalibration1(char*
input, char* output)
std::vector<Line>
                                  图像标定。
ImageCalibration2(char*
                                  需引入以下结构体:
                          input,
char* output)
                                  struct Line {
                                     double m, b;
                                     int dist_o; // distance to
                                  origin point when
                                         // line(x = dist o) is
                                  perpendicular to x axis
                                     int x0, x1, y0, y1; // two end
                                  points
                                     int end point num;
                                     Line (double _m, double _b,
                                  int dist o = 0, int x0 = 0, int
                                  _{y0} = 0,
                                         int x1 = 0, int y1 = 0,
                                  int _end_point_num = 0)
                                         :
                                                m(\underline{m}),
                                                            b(_b),
                                  dist o( dist o),
                                                          x0(x0),
                                  x1(_x1), y0(_y0), y1(_y1),
                                     end_point_num(_end_point_num
                                  ) {}
```

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 {
31260)	Point (int $x = 0$ , int $y = 0$ )
	{ this- $>$ x = x; this- $>$ y = y; }
	int x;
	int y;
	};
	struct CentersPoint {
	CentersPoint(Point point, int
	radius) { this->point = point;
	this->radius = radius; count =
	1; }
	Point point;
	int count;
	int radius;
	};
<pre>void ImgDisplay(char* input, char*</pre>	图像显示, input 是输入的图像文件
WindowsName)	名,WindowsName 是要创建的窗口名
	称。
vector <int> HoughTransform1(char*</int>	霍夫直线检测,返回检测结果。如果
input, char* output, unsigned char	共线线段之间的间隙小于fillGap,则
color[3], int numOfLines, int	连接它们,如果合并的行短于
fillGap, int minLength)	minLength,则放弃它,numOfLines是
	可能的线段数量, numOfLines=10,
	fillGap=5 , minLength=60 ,
	color={ 0, 255, 0 }。支持 8 位 BMP
	图像。
vector <int> HoughTransform2(char*</int>	霍夫直线检测,返回检测结果。如果
input, char* output, unsigned char	共线线段之间的间隙小于 fillGap,则
color[3], int numOfLines, int	连接它们,如果合并的行短于
fillGap, int minLength)	minLength,则放弃它,numOfLines 是
	可能的线段数量,numOfLines=10,
	fillGap=5 , minLength=60 ,
	color={ 0, 255, 0 }。支持8位BMP
	图像。
void SobelGradient(char*	用 sobel 算子分别求 x 方向, y 方向的
input, char* output)	梯度。
void SobelNoneMaxSpress(char*	用 sobel 算子分别求 x 方向, y 方向的
input, char* output)	梯度,再用梯度方向和非极大值抑制
	去掉冗余信息。
void HytheresisThresholding(char*	Hythereis 阈值处理, high=130,
input, char* output, double high,	low=60。
input, onar. output, doubte night,	1011 000

double low)	
·	霍夫变换, high=130, low=60,
char* output, double high, double	
low, double threshold)	
	霍夫变换, high=130, low=60,
char* output, double high, double	
low, double threshold)	threshold of the
vector <int> HoughTransfer2(char*</int>	霍夫变换,返回角点信息,以两个元
input, double high, double low,	素为一组。high=130, low=60,
double threshold)	threshold=0.4.
void FeatureDetection1(char*	特征检测。
input, char* output)	1寸11112199]。
	特征检测。
std::vector <sift::keypoint></sift::keypoint>	
FeatureDetection2(char* input,	需引入以下空间:
char* output)	namespace sift {
	struct Keypoint { //离散坐标系
	* "
	int i;
	int j;
	int octave;
	int scale; //倍频程内高斯图
	像的索引
	//连续坐标(内插)
	, , , , , , , , , , , , , , , , , , , ,
	float x;
	float y;
	float sigma;
	float extremum_val; //插值
	DoG 极值值
	-t-l., / .:
	std::array <uint8_t, 128=""></uint8_t,>
	descriptor;
	};
ALAMOUDD DOTAIN	上
std::vector <matched_point *=""></matched_point>	特征匹配。支持 PGM 图像。
FeatureMatching(char*	需引入以下结构体:
input1, char* input2, char* output)	<pre>typedef struct matchedpoint {</pre>
	unsigned int uiPosXFirst;
	unsigned int uiPosYFirst;
	unsigned int uiPosXSecond;
	unsigned int uiPosYSecond;
	bool bIsSample;
	boot bisbampie,

	}MATCHED POINT;
std::vector <std::pair<int, int="">&gt;</std::pair<int,>	特征匹配,返回匹配结果。
FeatureMatching1(char*	
input1, char* input2, char* output)	
std::vector <std::pair<int, int="">&gt;</std::pair<int,>	特征匹配,返回匹配结果。
FeatureMatching2(char* input1,	
char* input2, char* output)	
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 ,
	highThreshold=150。
void Canny(string input, string	Canny 算子,参考: output="output"。
output)	支持 BMP 文件。
<pre>void Canny(string input, char*</pre>	Canny 算子,参考: sigma=6.0,
output, float sigma, float	threshold=3.5。支持 BMP 文件。
threshold)	
void Hough(char* input, char*	霍夫变换, 参考: sigma=6.0,
output, float sigma, float	threshold=3.5, thre_val=0.5, color
threshold, double	用于设置画出的标定点和线的颜色,
thre_val, unsigned char* color)	如: color[3]={0,0,255}。支持 BMP 文
	件。
<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
222	件名。
void DES_Decrypt (char	DES 解密函数,支持多种文件。
*CipherFile, char *Key, char	CipherFile 是已加密文件的文件名,
*PlainFile)	Key 是密钥字符, PlainFile 是解密后
T1-4 W + 1 / 1	的文件名。
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
output, int channels, int ROTATION)	果图像文件名。channels 是图像的像     素通道数,返回值数组中的第一个元
	<u>新</u> 題題數,
	新定取入匹配分级,第二十九系定日     标的 X 轴坐标值,第三个元素是目标
	的 Y 轴坐标值,第三十九系定目标     的 Y 轴坐标值。至少支持 PNG 图像。
	対:抽坐体値。主グ文材 FNG 図像。   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
output, int channels, int ROTATION)	果图像文件名。channels 是图像的像
caspas, inc shamers, inc hornitoly	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。
	Nothinion ood Michilion 10

woid KMeans(string input, unsigned int Clusters, char* output)  void PGMSobel(char* input, char* output, int Mx[3][3], int My[3][3], int max, int min)  splitting type  woid PGMSobel(char* input, char* output, int Mx[3][3], int output, int Mx[3][3], int output, int max, int min)  splitting type  woid KMeans(string input, unsigned K-M Clu E4  void PGMSobel(char* input, char* output)  output, int Mx[3][3], int output)  splitting type  int output  output  output  output  int output  output  output  int output  output  output  int output  output  output  int output  outpu	leans 聚类, input 是输入文件名, sters 是聚类的种类数目, output 输出文件名。支持 BMP 文件。 el 算子, input 是输入文件名, put 是输出文件名。支持 P5 格式 PGM 文件。 be to the compact of the co
woid KMeans(string input, unsigned int Clusters, char* output)  void PGMSobel(char* input, char* output, int Mx[3][3], int Mx[3][3], int My[3][3], int max, int min)  参注	leans 聚类, input 是输入文件名, sters 是聚类的种类数目, output 输出文件名。支持 BMP 文件。 el 算子, input 是输入文件名, put 是输出文件名。支持 P5 格式 PGM 文件。  考模板:  Mx[3][3] = {{-1, 0, 1}, {-2, 2}, {-1, 0, 1}}  My[3][3] = {{-1, -2, -1}, {0, 0}, {1, 2, 1}}
woid KMeans(string input, unsigned int Clusters, char* output)  void PGMSobel(char* input, char* output, int Mx[3][3], int Mx[3][3], int My[3][3], int max, int min)  参 int 0, int	leans 聚类, input 是输入文件名, sters 是聚类的种类数目, output
woid KMeans(string input, unsigned int Clusters, char* output)  void PGMSobel(char* input, char* output, int Mx[3][3], int Mx[3][3], int My[3][3], int max, int min)  set	leans 聚类, input 是输入文件名, sters 是聚类的种类数目, output 偷出文件名。支持 BMP 文件。 el 算子, input 是输入文件名, put 是输出文件名。支持 P5 格式 PGM 文件。 考模板: Mx[3][3] = {{-1, 0, 1}, {-2, 2}, {-1, 0, 1}}
### type    State   State	leans 聚类,input 是输入文件名,sters 是聚类的种类数目,output
woid KMeans(string input, unsigned int Clusters, char* output)  void PGMSobel(char* input, char* soboutput, int Mx[3][3], int Mx[3][3], int 的参	leans 聚类,input 是输入文件名, sters 是聚类的种类数目,output 俞出文件名。支持 BMP 文件。 el 算子,input 是输入文件名, put 是输出文件名。支持 P5 格式 PGM 文件。 跨模板:
### type    State   State	leans 聚类,input 是输入文件名, sters 是聚类的种类数目,output 俞出文件名。支持 BMP 文件。 el 算子,input 是输入文件名, put 是输出文件名。支持 P5 格式 PGM 文件。
### typ    Styp	leans 聚类,input 是输入文件名, sters 是聚类的种类数目,output 偷出文件名。支持 BMP 文件。 el 算子,input 是输入文件名, put 是输出文件名。支持 P5 格式
需求 typ  } S  typ  void KMeans(string input, unsigned int Clusters, char* output)  Clu 是有  void PGMSobel(char* input, char* Sob	leans 聚类,input 是输入文件名, sters 是聚类的种类数目,output 输出文件名。支持 BMP 文件。 el 算子,input 是输入文件名,
需求 typ  } S  typ  void KMeans(string input, unsigned int Clusters, char* output)  Clu 是有	leans 聚类,input 是输入文件名, sters 是聚类的种类数目,output 俞出文件名。支持 BMP 文件。
需求 typ  } S  typ  void KMeans(string input, unsigned int Clusters, char* output)  Clusters	leans 聚类,input 是输入文件名, sters 是聚类的种类数目,output
需求 typ } S typ	
需求 typ } S typ	01,
需ityp	01:
需ityp	SSD colorD;
需ityp	int x, y;
需 <sup>i</sup> typ	edef struct solutionLocation {
需要	SD;
需要	int SSDr, SSDg, SSDb, sum;
	edef struct SumSquareD {
I ROI	引入以下结构体:
	ATION=360 或 ROTATION=1。
	更更致,返回匹配纪未致组。王少 寺 PNG 图像。参考:channels=3,
	到该文件名。Chamlers 定图像的像一 通道数,返回匹配结果数组。至少一
	Blatte 定件平图像,output 定结   图像文件名。channels 是图像的像
	以匹配, Input 走马本图像,   plate 是样本图像,output 是结
	版匹配, input 是母本图像,
	客。 多号: Channers-5, ATION=360或ROTATION=1。
	度目协的「抽坐协恒。主少义持「NG」像。参考: channels=3 ,
	是目标的Y轴坐标值。至少支持PNG
	一个几系定取入匹配分级,第二个一 素是目标的 X 轴坐标值,第三个元
	《的像系通道数,这回值数组中的   一个元素是最大匹配分数,第二个
1 - · · · · · · · · · · · · · · · · · ·	prate 是件本图像。channers 是   象的像素通道数,返回值数组中的
	双匹配, Input 走马本图像,   plate 是样本图像。channels 是
	版匹配, input 是母本图像,
	SSD colorD; ol;
	int x, y;
typ	edef struct solutionLocation {
	SD;
	int SSDr, SSDg, SSDb, sum;
typ	edef struct SumSquareD {
	引入以下结构体:

output, int Mx[3][3], int	output 是输出文件名。支持 P5 格式
My[3][3], int max, int min)	的PGM文件。
my [0] [0], The max, the min/	参考模板:
	$\inf_{x \in \mathbb{R}} Mx[3][3] = \{\{-1, 0, 1\}, \{-2, 1\}\}$
	$[0, 2], \{-1, 0, 1\}$
	int $My[3][3] = \{\{-1, -2, -1\}, \{0, \}\}$
	0, 0}, {1, 2, 1}}
	int max = -9999
	int min = 9999
void PGMSobelY(char* input, char*	Y 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
My[3][3], int max, int min)	的PGM文件。
My[3][3], IIIt max, IIIt milli/	
	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 PGMSobell(char* input, char*	Sobel 算子, input 是输入文件名,
output, int min, int max, int	output 是输出文件名。min 和 max 是
mx[3][3], int my[3][3])	图像归一化相关参数,如 min =
mx[0][0], iiit my[0][0]/	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}
	* / /
11 DOMO 1 191/ 1	
void PGMSobelX1(char* input, char*	X 方向梯度, input 是输入文件名,
output, int min, int max, int	output 是输出文件名。min 和 max 是
mx[3][3], int my[3][3])	图像归一化相关参数,如 min =
	1000000, max = 0; mx和 my分别是
	Sobel 算子的 X 和 Y 方向模板。支持
	P2 和 P5 格式的 PGM。
	参考模板:
	$\inf_{x \in [3]} \max[3] = \{$
	THE MALOJEOJ (

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

<pre>void HorizSobel(char* input, char* output)</pre>	水平 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 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 PGMSobel1(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 RAWSobelEdge(char* input, char* output, int ROWS, int COLS, int M, float sobelX[3][3], float sobelY[3][3])</pre>	Sobel 算子, input 是输入文件名, output 是输出文件名。ROWS 是图像的行, 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: float sobelX[3][3] = {{-1,0,1},
	float sobelY[3][3] = {{-1,-2,-1}, } {0,0,0},
void RAWPlaceHolder(char* input, char* output, int ROWS, int COLS, int M, float mask[3][3])	{1,2,1}}; 边缘检测, input 是输入文件名, output 是输出文件名。ROWS 是图像的 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: float mask[3][3] = {{-1,-2,-1}, {0,0,0},{1,2,1}};
<pre>void LaplacianEnhancement(char* input, char* output, int N, int LaplMask[3][3])</pre>	拉普拉斯图像增强, input 是输入文件名, output 是输出文件名。如: N=1。 支持 8 位 BMP 图像。 参考模板: int LaplMask[3][3] = {
void LaplaceSmooth(char*	拉普拉斯平滑, input 是输入文件名,

input, char* output, int N, int	output 是输出文件名。如: N=1。支持
<pre>input, char* output, int N, int LaplMask[3][3])</pre>	8位BMP图像。
Lapimask[3][3])	• • • • • • • • • • • • • • • • • • • •
	参考模板:
	int Lap1Mask[3][3] = {
	0, 1, 0,
	1, -4, 1,
	0, 1, 0
	};
void Sobell(char* input, char*	Sobel 算子, input 是输入文件名,
output, int N, int	output 是输出文件名。如: N=1。支持
Sb1Mask1[3][3], int	8 位 BMP 图像。
Sb1Mask2[3][3])	参考模板:
	int Sb1Mask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
	};
Cohol Cmooth (charate	
void SobelSmooth(char*	Sobel 平滑, input 是输入文件名,
input, char* output, int N, int	output 是输出文件名。如: N=1。支持
Sb1Mask1[3][3], int	8 位 BMP 图像。
Sb1Mask2[3][3])	参考模板:
	int Sb1Mask1[3][3] = {
	-1, -2, -1,
	0, 0, 0,
	1, 2, 1
	};
	int Sb1Mask2[3][3] = {
	-1, 0, 1,
	-2, 0, 2,
	-1, 0, 1
	};
void SobelOperator(char*	Sobel 算子, 耗时较长, input 是输入
input, char* output)	文件名, output 是输出文件名。支持
	24 位 BMP 图像。
SobelImage** SobelOperator(char*	返回处理后的各像素点的坐标和对应
input)	的像素值,若是边缘点则对应白色,
	否则对应黑色。支持 24 位 BMP 图像。
	需引入以下结构体:
	typedef struct {
1	cypouch schuck (

	1
	int x;
	int y;
	unsigned char red;
	unsigned char green;
	unsigned char blue;
	}SobelImage;
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。
<pre>void SURF(char* input1, char*</pre>	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 EdgeDetection(char*	边缘检测, input 是输入文件名,
input, char* output)	output 是输出文件名。支持 4 位 BMP
	图像。
void EdgeDetection1(char*	边缘检测, input 是输入文件名,
input, char* output, short	output 是输出文件名。支持 8 位 BMP
sharpen[3][3])	图像。
F F F	参考模板:
	short sharpen[3][3] = {{1, 1, 1},
	{1, -8,
	1},
	$\{1, \dots, 1, \dots\}$
	1}};
void EdgeDetection2(char*	
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
1 .,	图像像素的相关参数,如 a=3。支持 24
	位BMP图像。
void EdgeDetection4(char*	边缘检测, input 是输入文件名,
input, char* output, int a)	output 是输出文件名。a 是用于设置
	图像像素的相关参数,如 a=3。支持 24
	位 BMP 图像。
void YFiltering(char* input, char*	Y 方向滤波, input 是输入文件名,
output, int sobel x[3][3], int	output 是输出文件名。支持 P5 格式
sobel_y[3][3])	的 PGM 图像。
PODET TOTOTA	Harom 内は Mo
	参考構析.
	参考模板: int sobel_x[3][3] = { { 1, 0,

	`
	-1},
	{ 2,
	0, -2,
	{ 1,
	$0, -1\}\};$
	int $sobe1_y[3][3] = \{ \{ 1, \} \}$
	2, 1},
	$\{0,$
	$\{0, 0\},$
	{-1,
	$-2, -1\}\};$
void XFiltering(char* input, char*	X 方向滤波, input 是输入文件名,
output, int sobel_x[3][3], int	output 是输出文件名。支持 P5 格式
sobe1_y[3][3])	的 PGM 图像。
	参考模板:
	int sobe1 $_x[3][3] = \{ \{ 1, 0, \} \}$
	$-1\}$ ,
	$\{ 2,$
	[0, -2],
	$\{1,$
	$0, -1\}\};$
	int sobel_y[3][3] = { { 1,
	2, 1},
	{ 0,
	0, 0},
	{-1,
	$-2, -1\}$ ;
void SobelFiltering(char*	Sobel 算子,input 是输入文件名,
input, char* output, int	output 是输出文件名。支持 P5 格式
sobel_x[3][3], int sobel_y[3][3])	的 PGM 图像。
	参考模板:
	int sobel_x[3][3] = $\{ \{ 1, 0, \} \}$
	-1},
	{ 2,
	$0, -2\},$
	$\{ 1, 0, -1 \} \};$
	int sobel_y[3][3] = { { 1,
	2, 1},
	{ 0,
	0, 0,
	{-1,
	$-2, -1\}$ ;
void 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
prewitt_x[3][3], int	的 PGM 图像。
prewitt_y[3][3])	参考模板:
	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, -
	$\{5, -3\}\};$
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
laplacian[3][3])	的 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}};
void SobelOperation1(char*	Sobel 算子,支持 BMP 图像。
input, char* output, int width, int	
height)	
void SobelOperation2(char*	Sobel 算子,支持 BMP 图像。
input, char* output, int width, int	
height)	N. Maria Nella Indiana FET Ve
void Roberts(char* input, char*	Roberts 边缘检测,支持 BMP 图像。
output)	
void Prewitt(char* input, char*	Prewitt 边缘检测,支持 BMP 图像。
output)	Calcal 计模较测量 去柱 pun 网络
void Sobel(char* input, char*	Sobel 边缘检测,支持 BMP 图像。
output)	Lanlace 边络松洲 古共 DVD 图像
<pre>void Laplace(char* input, char* output)</pre>	Laplace 边缘检测,支持 BMP 图像。
void BoxBlurAdvanced(string	   高级方框模糊,参考: radius=5。支持
input, string output, int radius)	同级力性疾物,参考:Faurus-5。又持一PNG文件。
void HoughTransform(char*	霍夫变换, input 是输入的 RAW 文件,
input, char* output, unsigned char	output 是输出的 RAS 文件,
input, chair output, unsigned chai	output 定相田田 MAS 文件,

	,
threshold, int rhol, int	threshold=100, a=800, b=255, rho1=-
thetal, int rho2, int theta2, int	295 , theta1=129 , rho2=-171 ,
rho3, int theta3, int a, unsigned	theta2=51, rho3=312, theta3=14.
char b)	
static void	边缘检测,参考: a=0.33, b=0.33,
EdgeDetectionWithoutNonmaximum(co	c=0.33。支持24位BMP图像。
nst LPCTSTR input, const LPCTSTR	
output, double a, double b, double	
c)	
static void	边缘检测,参考:orank=20,oranb=80。
CannyEdgeDetection(const LPCTSTR	支持 24 位 BMP 图像。
input, const LPCTSTR	7.11 E 2.11 E 3.11
output, double a, double b, double	
c, int orank, int oranb)	
static void HoughTransform(const	霍夫变换,参考: a=0.33, b=0.33,
_	(c=0.33, orank=20, oranb=80。 支持 24
- '	
output, double a, double b, double	位 BMP 图像。
c, int orank, int oranb)	其加子長葉州 十共 DNO 文件
void BoxBlurBasic(string	基础方框模糊,支持 PNG 文件。
input, string output)	
void SobelSharpen(char*	Sobel 算子, input 是输入文件名,
input, char* output, int	output 是输出文件名。Templatex 是
Templatex[3][3], int	laplace 锐化模板,4 邻域,Templatey
Templatey[3][3], int	是 laplace 锐化模板,8 邻域,
coefficient1, int coefficient2)	coefficient1 = 9, coefficient2 =
	16。支持 24 位 BMP 图像。
void EncryptionDecryption(char*	图像加解密,Key 是密钥,a=1 时执行
input, char* output, int Key, int a)	加密,a=0 时执行解密。支持 24 位 BMP
	图像。
<pre>void Encryption(char* input, char*</pre>	图像加密, input 是输入文件名,
output, int Key)	output 是输出文件名。Key 是密钥。
	支持 24 位 BMP 图像。
void Decryption(char* input, char*	图像解密, input 是输入文件名,
output, int Key)	output 是输出文件名。Key 是密钥。
	支持 24 位 BMP 图像。
void Nesting(char* Biginput, char*	图像嵌套, Biginput 是输入的大图,
Smallinput, char* output)	Smallinput 是输入的小图。支持 24 位
	BMP 图像。
void Blend(char* input1, char*	图像融合之混合化, input1 和 input2
input2, char* output)	是输入的两个要融合的图像,output
	是输出文件名。支持 24 位 BMP 图像。
void Checker (char* input1, char*	图像融合之棋盘化, input1和 input2
input2, char* output)	是输入的两个要融合的图像,output
inputa, onar · output/	是输出文件名。支持 24 位 BMP 图像。
	化制山入口石。又对 25 世 DMI

void Blend1(char* input1, char*	图像融合之混合化, input1 和 input2
input2, char* output)	是输入的两个要融合的图像, output
	是输出文件名。支持 24 位 BMP 图像。
void Checkerl(char* input1, char*	图像融合之棋盘化, input1 和 input2
input2, char* output)	是输入的两个要融合的图像,output
imputz, char · output/	是输出文件名。支持 24 位 BMP 图像。
· 1	
void QRCodeGeneration(char	二维码生成,filename 是生成的二维
*filename, char* inputString)	码图像文件名,inputString是二维码
	包含的信息。支持 BMP 图像。
vector <float></float>	角点检测,支持 P5 和 P6 格式的 PPM
HarrisCornerDetection(char*	文件,从返回值数组中的第一个元素
input, int width, int height, int	开始,返回值以3个元素为一组,分
channels, int step, float	别是角点的 X 坐标、Y 坐标和分数,
threshold, float k, float sigma)	若返回值数组名为 A ,则
threshord, froat K, froat Signa)	·
	{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,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:
(21 ) 1 1 2 (1	threshold=2000, k=1, sigma=1.2.
vector <float> HarrisCorner(char*</float>	角点检测,支持 P5 和 P6 格式的 PPM
input, char* output, int width, int	文件,从返回值数组中的第一个元素
height, int channels, float	开始,返回值以 3 个元素为一组,分
threshold, float k, float sigma)	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°
int* TemplateMatching(char*	模板匹配,返回值中第一个元素是匹
input1, char* input2, char*	配框左上角的纵坐标,第二个元素是
output, unsigned char red, unsigned	匹配框左下角的纵坐标,第三个元素

是匹配框的左上角的横坐标, 第四个 char green, unsigned 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。支持 BMP 图像。 vFerestre 模板匹配,返回目标的位置等数据。 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 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; TemplateMatching(char\* 模板匹配,返回值中第一个元素是匹 input1, char\* input2, unsigned char 配框左上角的纵坐标,第二个元素是 red, unsigned char green, unsigned 匹配框左下角的纵坐标, 第三个元素 是匹配框的左上角的横坐标,第四个 char blue, double MatchScore) 元素是匹配框的右上角的横坐标。 input1 是搜索图像, input2 是模板图 像, output 是匹配结果图像, MatchScore=0.9。支持 BMP 图像。 TemplateMatching(char\* 模板匹配, suprapunereMaxima表示最 void input, char\* 大 templatename, char\* 重 叠 率 考 output, unsigned MaximumMatchingQuantity=10 int MaximumMatchingQuantity, double MatchScore=0.8 MatchScore, float suprapunereMaxima=0.2。支持 BMP 图 suprapunereMaxima, unsigned 像。 red, unsigned char green, unsigned char blue) 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)。支持 BMP 图像。 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。 需引入一下结构体: input1, char\* input2, char\* output) struct coord {

```
int x = 0;
                                 int y = 0;
                             模板匹配,返回值是匹配框的左上角
int*
        TemplateMatching4(char*
                             坐标(x,y)。支持 BMP 图像。
input1, char* input2)
                             模板匹配, 返回值是匹配框的左上角
vector(coord)
                             坐标组成的向量。
TemplateMatching3(char*
input1, char* input2)
                             需引入一下结构体:
                             struct coord {
                                 int x = 0;
                                 int y = 0;
                             模板匹配,返回值是匹配框的左上角
vector<int>
TemplateMatching(char*
                             坐标(x, y)组成的向量,按顺序以两
input1, char* input2, float min)
                             个为一组。min 是与匹配分数相关的
                             参数,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
TemplateMatch(char*
                   inputfile,
                             点的横坐标和纵坐标,第3个元素是
char* templatefile, char* output,
int size, int best loss, double a,
                             目标与模板相对的旋转角度,第4个
double b, double c, double d, int
                             元素是缩放比例。参考: size=1,
e1, int e2)
                             best loss=10000000000 , 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;
vector<TargetData>
                             模板匹配,支持 JPG 图像,返回值的
TemplateMatch(char*
                             第1和第2个元素是匹配框左上角顶
                   inputfile,
char* templatefile, int size, int
                             点的横坐标和纵坐标,第 3 个元素是
                             目标与模板相对的旋转角度,第4个
best loss, double a, double b,
                             元素是缩放比例。参考: size=1,
double c, double d, int el, int
e2)
                             best loss=10000000000 , 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;
```

int ObjectFind(char\* inputfile,
char\* Templatefile)

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,返回匹配到的目标数量。 支持 BMP 图像。

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: typedef struct Point { int x; int y;

}Point;

double\* TemplateMatching1(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>

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\_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\* 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\_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;

}Point;

int y;

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\*

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 inputFile, char\* templateFile, char\* output txt, double threshold, unsigned char color, unsigned char red, unsigned char green, unsigned char blue)

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\* templateFile, 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。

TemplateMatching3(char\* inputFile, char\* templateFile, unsigned char red, unsigned char green, unsigned char blue, double

c, double threshold, int a, int

模板匹配,返回以匹配框的中心点坐 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, threshold=0.9, a=20, b1=-30, b2=30, c1=5, c2=20。支持 PPM 文件。

vector<int> TemplateMatching(char\* input, char\* Template)

b1, int b2, int c1, int c2)

vector <double>

模板匹配,支持24位BMP图像,返回 以匹配框的左上角坐标(x,y)组成的 按顺序存储的数组。

int FeatureDetectionNumber(char\* input, char\* output, float a, int b, float red, float green, float blue)

特征检测,返回找到的特征点的数量。 a 是亮度值,如 a=255, b 与图像清晰 度有关,默认 b=100; red、green、blue 分别是特征点指示器颜色的红色、绿 色、蓝色的通道值,如red=0,green=0, blue=1。支持 PPM 文件。

std::vector<sift::Keypoint> FeatureDetection(char\* input, char\* output, float a, int | 度有关, 默认 b=100; red、green、blue

特征检测,返回找到的特征点的数量。 a 是亮度值,如 a=255, b 与图像清晰

```
分别是特征点指示器颜色的红色、绿
b, float red, float green, float
blue)
                               色、蓝色的通道值,如red=0,green=0,
                               blue=1。支持 PPM 文件。
                               需引入以下命名空间:
                               namespace sift {
                               struct Keypoint {
                                   int i;
                                   int j;
                                   int octave:
                                   int scale;
                                   float x;
                                   float y;
                                   float sigma;
                                   float extremum_val;
                                   std::array<uint8_t,</pre>
                                                         128>
                               descriptor:
                               };
                               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)
std::list<sift::SiftKeypoint>
                               特征检测,返回特征点检测结果。支
                               持 PPM 文件。
FeatureDetection(char*
input, char* output, unsigned char
                               需引入以下结构体:
                               SIFT 关键点: 128 维
red, unsigned char green, unsigned
char blue)
                               namespace sift {
                               #define
                                         DEGREE OF DESCRIPTORS
                               (128)
                               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
```

// 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\* 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, input1, char\* input2, char\* 第2个值和第3个值分别是是图像1 output, unsigned char red, unsigned char green, unsigned char blue) 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。 特征匹配,返回找到的特征匹配点的 FeatureMatching(char\* 数量。a 是亮度值,如 a=255, b 与图 input1, char\* input2, char\* output, 像清晰度有关,默认 b=100; red、 float a, int b, float red, float green、blue 分别是匹配线颜色的红 green, float blue) 色、绿色、蓝色的通道值,如 red=0, green=0, blue=1。支持 PPM 文件。 unsigned int 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。 FeatureMatching1 (char\* input 是输入图像, output 是生成的 input1, char\* input2, char\* output, unsigned char red, unsigned 特 征 点 冬 像 bExtractDescriptor=true。返回匹配 char green, unsigned char 的特征点数量。支持 PGM 文件。 blue, bool bExtractDescriptor) vector<int> 返回值中的数据格式为: 第1个值是 FeatureMatching2(char\* 特征对的序号,从0开始,第2个值 input1, char\* input2, char\* 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 output, unsigned char red, unsigned green, unsigned 值和第5个值分别是是图像2的其中 char char 一个特征点的横坐标和纵坐标并与第 blue, bool bExtractDescriptor) 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 返回特征点数量。input 是输入图像, int output 是生成的特征点图像, FeatureExtraction1(char\* input, char\* output, unsigned char bExtractDescriptor=true。支持 PGM red, unsigned char green, unsigned 文件。 char blue, bool bExtractDescriptor) 返回特征点列表。input 是输入图像, std::list<ezsift::SiftKeypoint> output 是生成的特征点图像, FeatureExtraction2(char\* input, char\* output, unsigned char bExtractDescriptor=true。支持 PGM red, unsigned char green, unsigned 文件。 需引入以下命名空间: char blue, bool bExtractDescriptor) namespace ezsift { #define DEGREE (128) //SIFT 关 键点: 128 维 struct SiftKeypoint { int octave; // octave 数量 // layer 数量 int layer; float rlayer; // layer 实际数 // 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 descriptors[DEGREE]; float //描述符 }; 查找缺陷位置,返回材料缺陷的位置, vector<int> DefectLocation(char\* TemplateFile, char\* SampleFile, int 以每4个元素为一组。TemplateFile 是模板图像,SampleFile是样本图像, floor, int size, int a, int b, int g 是缺陷的有效界限的 X 轴长度, h 是 c, int d, int e, int f, int g, int 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\* 缺陷尺寸,返回缺陷尺寸,以4个为 TemplateFile, char\* SampleFile, int 一组。Template 是模板图像,Sample 是样本图像, g 是缺陷的有效界限的 X floor, int size, int a, int b, int 轴长度, h 是缺陷的有效界限的 Y 轴 c, int d, int e, int f, int g, int h, int FULL, int EMPTY, bool report) 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。 样品好坏的数量,返回结果中,第一 vector<int> GoodBadQuantity(char\* 个元素是合格的圆圈数量, 第二个元 TemplateFile, char\* SampleFile, int floor, int size, int a, int b, int 素是缺陷的圆圈数量。Template 是模 c, int d, int e, int f, int g, int 板图像, Sample 是样本图像, g 是缺 陷的有效界限的 X 轴长度, h 是缺陷 h, int FULL, int EMPTY, bool report) 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。 圆检测, 返回找到的圆的位置和大小 hough\_param\_circle\* 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
hough param circle*>
                              文件。
CircleDetection2(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,返回圆心的坐标和半径,返
unsigned
                         int*
                              回值数组中第一个元素是圆心 X 坐
CircleDetection1(char* input)
                              标,第二个元素是圆心 Y 坐标,第三
                              个元素是圆半径。支持 BMP 图像。
VECTORS
         CircleDetection2(char*
                              圆检测, 返回圆心的坐标和半径, 返
input)
                              回以圆心 X 坐标, 圆心 Y 坐标, 圆半
                              径组成的结构体。
                              需引入以下结构体:
                              typedef struct {
                                 int* x;
                                 int* y;
                               unsigned int size;
                               VECTORS;
SWARM*
         CircleDetection3(char*
                              圆检测,返回圆心的坐标和半径,返
input)
                              回圆检测结果。
                              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
	typeder struct
	unsigned int nParticles;//
	Número de PARTICLES del SWARM
	unsigned int nParams;
	unsigned int idGbest; //
	índice 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> &gt;</vector<float>	圆检测,返回圆心的坐标和半径,返
CircleDetection4(char* input)	回圆检测结果。
int Equal(char* input1, char*	若比对图像的梯度幅相似性偏差值等
input2, double c)	于 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
input1, char* input2, double c)	于 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 LessThan(char* input1, char*	若比对图像的梯度幅相似性偏差值小
input2, double c)	于 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。
double GMSD(char* input1, char*	求两幅图像的梯度幅相似性偏差值并
input2)	返回结果。input1 和 input2 是要比
•/	对的两个图像。支持 24 位 BMP 图像。
vector <vector<double>&gt;</vector<double>	图像标定,返回各个顶角的坐标。参
Correction(string input, char*	考: A_height = 297, A_width = 210,
output, int A height, int	thre val=0.5 , sigma=6.0 ,
A_width, double thre_val, double	threshold=3.5, angle num=180,
sigma, double threshold, int	range thre=5, fun thre=0.8,
angle_num, int range_thre, double	point_thre=3。支持 BMP 文件。
angre_num, int range_time, doubte	botur-mie oo X14 pm XII o

fun_thre, int point_thre)	
vector <double></double>	图像标定,返回各个直线的交点的 X
CalibrationAndCorrection(char*	轴和Y轴坐标,若以vector <double></double>
input, char* output1, char*	p 接收返回值则交点 1 的坐标为
output2, double sigma, double	(p[0], p[1]), 交点 2 的坐标为
gra_threshold, double	(p[2],p[3]),以此类推。参考:
vote_threshold, double	sigma=5.5 , gra_threshold=30 ,
peak_dis, unsigned char*	vote_threshold=1000 ,
lines_color, unsigned char*	peak_dis=200 ,
intersections_color)	lines $color[3] = \{0, 255, 0\}$
_	intersections $color[3] = \{0, 0, 255\}$
	。至少支持 BMP 图像。
float* ImageMatching(char*	图像匹配,返回值中的前10个元素是
TargetImage, char* TemplateO, char*	按顺序对应的目标与模板的差异分
	数,最后一个元素是匹配到的模板的
Template3, char* Template4, char*	序号。支持 BMP 图像。
Template5, char* Template6, char*	
Template7, char* Template8, char*	
Template9)	
float* ImageMatching(char*	图像匹配,返回值中的前 2 个元素是
TargetImage, char* Template0,	按顺序对应的目标与模板的差异分
char* Template1)	数,最后一个元素是匹配到的模板的
	序号。支持 BMP 图像。
void ImageFeatures(char*	图像特征。
input, char* kernel, char* output)	kernel 文件内容样例:
	3
	1
	0 -1 0
	-1 5 -1
	0 -1 0
	其中,3表示尺寸为3*3,1表示内核
11 D11 W 12 / 1 D20 /	的规模
void   FileWrite(char*   BMP, char*	图像隐写之文件写入,将文本文件写
TXT)	入图像。支持 32 位 BMP 图像。BMP 是
	要写入的图像文件名,TXT 是要写入
	图像的文本文件名。
<pre>void FileWriteOut(char* BMP, char*</pre>	图像隐写之文件写出,将文本文件从
TXT)	图像中取出来。支持 32 位 BMP 图像。
	BMP 是要写出的图像文件名, TXT 是写
	出图像后信息保存的文本文件名。
void LBP(char* input, char*	LBP 图像特征提取。支持 PNG 图像。
output)	20. 国际自由(A) V11 III 国际。
void LBP(char* input, char*	LBP 图像特征提取。radius 是取样半
output, int choice, float	径(不小于 1 的浮点数),

radius, int pointNumbers)	pointNumbers 是取样点数(不小于 8 的整数),choice 的不同取值表示: 1. 普通圆形 LBP 2. 旋转不变圆形 LBP 3. 等价模式圆形 LBP。支持 24 位 BMP 图 像 , 只 有 等 价 模 式 下 取 样 点 pointNumbers 可以大于 8 但不得大于 255。
<pre>void Watershed2(char* input, char*</pre>	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)	图像。
void EcrireImage1(char*	图像分割。rayon=5。支持 PNG 图像。
input, char* output, uint32 t	-
rayon)	
void EcrireImage2(char*	图像分割。rayon=5。支持 PNG 图像。
input, char* inputMarqueurs, char*	
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)	
<pre>void Watershed1(char* input, char*</pre>	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*	
output, uint16_t rayon)	
void	图像分割。rayon=1。支持 PNG 图像。
EcrireImageCouleursAleatoires(cha	
r* input, char*	
inputMarqueurs, char*	
output, uint8_t r, uint8_t	
g, uint8_t b, uint16_t rayon)	
void Watershed(char* input, char*	图像分割之分水岭算法。
inputMarqueurs, char*	inputMarqueurs 是输入图像的标记图
output, uint8_t r, uint8_t	像。a 一般为 255, rayon=1。支持 PNG
g, uint8_t b, uint8_t a, uint16_t	图像。
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	国家分割之侵小填充法。x-0,y-0, novaCor=127。支持 PGM 文件。
novaCor)	110 V (10 O) 12 1 0 X (17 1 O) 11 X (17 0
void	读取点坐标并输出点之间绘制的点或
ConvertCoordinatesToGraphics(char	线段的图像。IMAGE SIZE=800,
* input, char* output, double	PIXEL_PADDING=25, drawlines=0 或
IMAGE SIZE, double PIXEL PADDING,	drawlines=1.
Initial_SIZE, doubte I INDE_I INDUINO,	GEGHTIHOD IV

	46 > ), b) 16 B
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
	1 x 坐标 y 坐标
	2 x 坐标 y 坐标
	3 x 坐标 y 坐标
	4 x 坐标 y 坐标
	N x 坐标 y 坐标
void HumanDetection1(char*	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)	) (1,1 = 1 E 2
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
input)	BMP 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
NN SQ DIST RATIO THR)	大数量, NN SQ DIST RATIO THR 是 NN
	和第二个 NN 之间距离平方比的阈值,
	参考: KDTREE BBF MAX NN CHKS=100,
	NN SQ DIST RATIO THR=0.49。支持24
	位 BMP 图像。
Feature* ImageFeature(char*	返回图像的特征数据。支持 24 位 BMP
input)	图像。
Tipa ()	需引入一下结构体:
	typedef struct Point2D64f
	{
	double x;
	double y;
	} Point2D64f;
	), omeabout,
	typedef struct feature
	{
	double x;
	/x 坐标/
	double y;
	y,

	/y 坐标/
	·
	/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;
void QRCodeGenerate(string	二维码生成, inputString 是二维码的
inputString, char* output)	内容, output 是输出的二维码图像文
	件名,支持BMP图像。
detectie* TemplateMatchingl(char*	模板匹配,返回各个目标的位置等数
input, char* Template, char*	据, 支持 24 位 BMP 图像。
output, float MatchScore, pixel	MatchScore=0.6, k=0.
C, unsigned int k)	需引入以下结构体:
c, and igned int h	typedef struct { unsigned char R;
	unsigned char unsigned char
	_
	G;
	unsigned char
	B;}pixel;

detectie** TemplateMatching2(char* input, char* Template, char* output, float MatchScore, pixel C, unsigned int k)	typedef struct { unsigned int x; unsigned int y; float prag; pixel culoare; } detectie;  模板匹配,返回各个目标的位置等数据,支持 24 位 BMP 图像。 MatchScore=0.6, k=0。 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 y; float prag; pixel
	culoare; }detectie;
void StyleTransfer(char*	图像风格迁移,至少支持 JPG 和 PNG
<pre>input1, char* input2, char* output, int iterations)</pre>	图像。iterations=10。
void ImageFusion(char*	图像融合,支持多种图像格式。
input1, char* input2, char*	
inputUniqe1, char*	
inputUniqe2, char* output)	図偽融合 古共夕研図偽技士 D_055
<pre>void Uniqe(char* input, char* inputUniqe, char* output, double R, double G, double B)</pre>	图像融合,支持多种图像格式。R=255, G=0, B=0。
<pre>void AVGFused(char* input1, char* input2, char* output)</pre>	图像融合,支持 PGM 图像。
void MAXFused(char* input1, char* input2, char* output)	图像融合,支持 PGM 图像。
void MINFused(char* input1, char* input2, char* output)	图像融合,支持 PGM 图像。
void FreqFused(char* input1, char* input2, char* output)	图像融合,支持 PGM 图像。
double CharacterRecognition(char*	字符匹配,支持 BMP 图像,返回值是

TargetImage, char*	目标图像匹配到的模板文件的序号,
TemplateFileGroup[])	如返回值是 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
CharacterRecognition1(char*	目标图像匹配到的模板文件的序号,
TargetImage, char*	如返回值是 2 则说明图像与序号为 2
TemplateFileGroup[])	(序号从零开始)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" };