

**算子库使用手册**

|  |  |
| --- | --- |
| 作者：- | **SINSEGYE** |
| 职务：技术工程师 |
| 邮箱：-@Sinsegye.com.cn | 北京市 朝阳区 |
| 服务热线：400-013-2158 | 望京东金辉大厦1801 [100102] |

**版权声明**

本手册的所有权归中科时代计算机系统有限公司所有。未经书面许可，任何人不得以任何形式复制、分发、翻译或以其他方式使用本手册的全部或部分内容。

本手册受版权法保护。任何对本手册内容的复制、分发、翻译、展示、表演、演绎或使用，无论出于何种目的，均需得到中科时代计算机系统有限公司的明确许可。未经许可，任何行为均视为侵犯中科时代计算机系统有限公司的版权。

中科时代计算机系统有限公司保留随时更新本手册的权利，并在不通知的情况下进行更新。对于因使用或信赖本手册所载明或未明示的信息而造成的任何损失或损害，中科时代计算机系统有限公司不承担任何责任。

在使用本手册时，您必须遵守所有适用的国家法律和法规。不得使用本手册进行非法活动或违反任何道德准则。

目录

[一、 接口说明 4](#_Toc25586)

[（一） 常用数据类型（连峰） 5](#_Toc16289)

[（二） 相机服务算子库（刘涛） 6](#_Toc15344)

[（三） 视觉基础算子库（连峰） 7](#_Toc19035)

[1． 图像操作 7](#_Toc16580)

[2． 图像绘制 9](#_Toc13689)

[3． 图像处理 11](#_Toc22713)

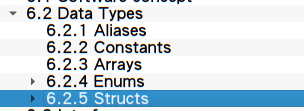
[4． 图像运算 26](#_Toc3917)

[（四） 时序算子库（昊庭） 27](#_Toc23875)

# 接口说明

本章主要描述了常用数据类型、相机服务算子库、基础视觉算子库和时间序列算子库在MateFacture中调用的接口。这些接口提供了一种可通过st脚本在PLC工程中实现对相机的操作以及对图像和时间序列的处理。

## 常用数据类型（连峰）



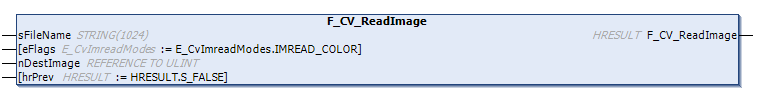
参考twincat vision的6.2.3到6.2.5节

## 相机服务算子库（刘涛）

## 视觉基础算子库（连峰）

### 图像操作

#### F\_CV\_ReadImage



读取图像，从文件加载图像，目前，支持以下文件格式：

* Windows位图: \*.bmp, \*.dib;
* JPEG文件: \*.JPEG, \*.jpg, \*.jpe;
* JPEG 2000文件: \*.jp2;
* 便携式网络图形: \*.png;
* WebP: \*.WebP;
* 便携式图像格式: \*.ppm, \*.pgm, \*.ppm\*.pxm, \*.pnm;
* PFM文件: \*.PFM;
* 太阳光栅: \*.sr, \*.ras;
* TIFF文件: \*.TIFF, \*.tif;
* OpenEXR图像文件: \*.exr;
* 辐射HDR: \*.HDR，\*.pic;

**Syntax**

Definition:

FUNCTION F\_CV\_ReadImage: HRESULT

VAR\_INPUT

sFileName : STRING(1024);

eFlags : E\_CvImreadModes := E\_CvImreadModes.IMREAD\_COLOR;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

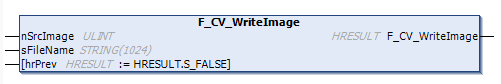
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| sFilename | STRING | 要加载的文件的名称 |
| eFlags | E\_CvImreadModes | 图像解码类型   1. IMREAD\_UNCHANGED: 解码不变 2. IMREAD\_GRAYSCALE: 解码灰度 3. IMREAD\_COLOR: 解码颜色 |
| nDestImage | REFERENCE TO ULINT | ram中加载的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_WriteImage



保存图像，将图像保存到指定文件，一般来说只能保存8位单通道或3通道（具有“BGR”通道顺序）图像，但以下情况除外：

* 16位无符号（16U）图像可以保存为PNG、JPEG 2000和TIFF格式；
* 32位浮点（32F）图像可以保存为PFM、TIFF、OpenEXR和Radiance HDR格式；

将使用LogLuv高动态范围编码（每像素4字节）保存3通道（32FC3）TIFF图像；使用此功能可以保存带有阿尔法通道的PNG图像。为此，创建8位（或16位）4通道图像BGRA，阿尔法通道位于最后。完全透明的像素的阿尔法应设置为0，完全不透明的像素的阿尔法应设置为255/65535。

**Syntax**

Definition:

FUNCTION F\_CV\_WriteImage: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

sFileName : STRING(1024);

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

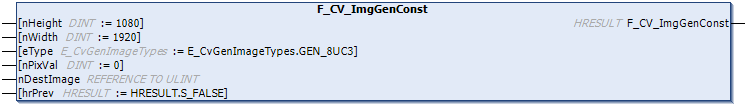
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | ram中加载的图像 |
| sFilename | STRING | 文件的名称 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgGenConst



数据生成图像，指定一个像素值和图片类型，生成一张图片，图片中的像素值全部为指定的值。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgGenConst: HRESULT

VAR\_INPUT

nHeight : DINT := 1080;

nWidth : DINT := 1920;

eType : E\_CvGenImageTypes := E\_CvGenImageTypes.GEN\_8UC3;

nPixVal : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nHeight | DINT | 生成图像的高度 |
| nWidth | DINT | 生成图像的宽度 |
| eType | E\_CvGenImageTypes | 生成图像的类型，枚举如下 |
| nPixVal | DINT | 生成图像的指定像素值 |
| nDestImage | REFERENCE TO ULINT | 生成的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

图像类型

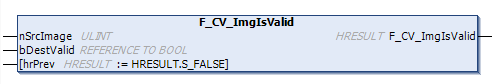
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GEN\_8UC1 | GEN\_8SC1 | GEN\_16UC1 | GEN\_16SC1 | GEN\_32SC1 | GEN\_32FC1 | GEN\_64FC1 | GEN\_16FC1 |
| GEN\_8UC2 | GEN\_8SC2 | GEN\_16UC2 | GEN\_16SC2 | GEN\_32SC2 | GEN\_32FC2 | GEN\_64FC2 | GEN\_16FC2 |
| GEN\_8UC3 | GEN\_8SC3 | GEN\_16UC3 | GEN\_16SC3 | GEN\_32SC3 | GEN\_32FC3 | GEN\_64FC3 | GEN\_16FC3 |
| GEN\_8UC4 | GEN\_8SC4 | GEN\_16UC4 | GEN\_16SC4 | GEN\_32SC4 | GEN\_32FC4 | GEN\_64FC4 | GEN\_16FC4 |
| GEN\_8UC5 | GEN\_8SC5 | GEN\_16UC5 | GEN\_16SC5 | GEN\_32SC5 | GEN\_32FC5 | GEN\_64FC5 | GEN\_16FC5 |
| GEN\_8UC6 | GEN\_8SC6 | GEN\_16UC6 | GEN\_16SC6 | GEN\_32SC6 | GEN\_32FC6 | GEN\_64FC6 | GEN\_16FC6 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgIsValid



判断输入图像是否有效，如果输入的图像为空，则为无效，反之则为有效。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgIsValid: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

bDestValid : REFERENCE TO BOOL;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

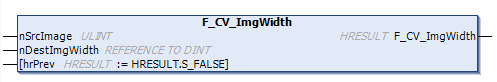
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| bDestValid | REFERENCE TO BOOL | 图像是否有效，true为有效，false为无效 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgWidth



获取输入图像的宽。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgWidth: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImgWidth : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

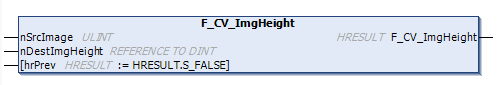
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nDestImgWidth | REFERENCE TO DINT | 图像宽 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgHeight



获取输入图像的高。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgHeight: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImgHeight : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

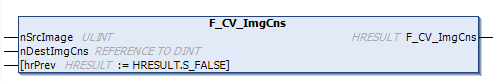
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nDestImgHeight | REFERENCE TO DINT | 图像高 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgCns



获取输入图像的通道数。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgCns: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImgCns : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

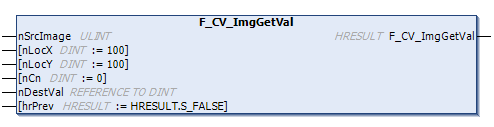
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nDestImgCns | REFERENCE TO DINT | 图像通道数 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgGetVal



获取输入图像中指定位置的像素值，给定图像中的坐标与通道号。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgGetVal: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nLocX : DINT := 100;

nLocY : DINT := 100;

nCn : DINT := 0;

nDestVal : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

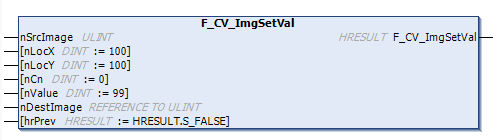
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nLocX | DINT | 获取像素值的x坐标 |
| nLocY | DINT | 获取像素值的y坐标 |
| nCn | DINT | 获取像素值的通道号 |
| nDestVal | REFERENCE TO DINT | 获取的像素值 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgSetVal



修改输入图像中指定位置的像素值，给定图像中的坐标与通道号。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgSetVal: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nLocX : DINT := 100;

nLocY : DINT := 100;

nCn : DINT := 0;

nValue : DINT := 99;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

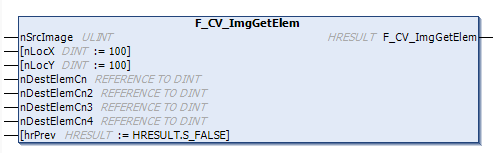
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 待修改的图像 |
| nLocX | DINT | 修改像素值的x坐标 |
| nLocY | DINT | 修改像素值的y坐标 |
| nCn | DINT | 修改像素值的通道号 |
| nValue | DINT | 要修改的像素值 |
| nDestImage | REFERENCE TO ULINT | 修改像素值后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgGetElem



获取图像中指定位置的所有通道的像素值，最多支持4通道，给定图像中的坐标。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgGetElem: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nLocX : DINT := 100;

nLocY : DINT := 100;

nDestElemCn : REFERENCE TO DINT;

nDestElemCn2 : REFERENCE TO DINT;

nDestElemCn3 : REFERENCE TO DINT;

nDestElemCn4 : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

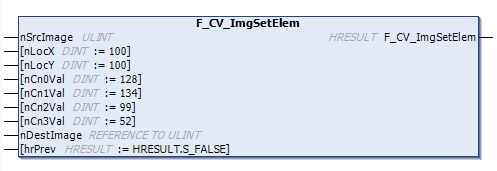
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nLocX | DINT | 获取像素值的x坐标 |
| nLocY | DINT | 获取像素值的y坐标 |
| nDestElemCn | REFERENCE  TO DINT | 在第1个通道获取的像素值，若该通道无像素值则为0 |
| nDestElemCn2 | REFERENCE  TO DINT | 在第2个通道获取的像素值，若该通道无像素值则为0 |
| nDestElemCn3 | REFERENCE  TO DINT | 在第3个通道获取的像素值，若该通道无像素值则为0 |
| nDestElemCn4 | REFERENCE  TO DINT | 在第4个通道获取的像素值，若该通道无像素值则为0 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgSetElem



修改图像中指定位置的所有通道的像素值, 最多支持4通道，给定图像中的坐标。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgSetElem: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nLocX : DINT := 100;

nLocY : DINT := 100;

nCn0Val : DINT := 128;

nCn1Val : DINT := 134;

nCn2Val : DINT := 99;

nCn3Val : DINT := 52;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

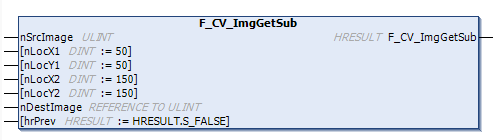
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 待修改的图像 |
| nLocX | DINT | 修改像素的x坐标 |
| nLocY | DINT | 修改像素的y坐标 |
| nCn0Val | DINT | 需要修改的第1个通道的像素值 |
| nCn1Val | DINT | 需要修改的第2个通道的像素值 |
| nCn2Val | DINT | 需要修改的第3个通道的像素值 |
| nCn3Val | DINT | 需要修改的第4个通道的像素值 |
| nDestImage | REFERENCE TO ULINT | 修改后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgGetSub



获取子区域，通过定义矩形的左上角与右下角的坐标，在输入图像中获取矩形子区域。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgGetSub: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nLocX1 : DINT := 50;

nLocY1 : DINT := 50;

nLocX2 : DINT := 150;

nLocY2 : DINT := 150;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

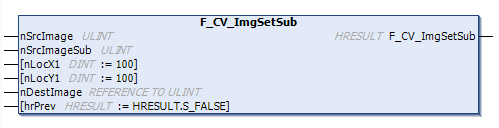
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nLocX1 | DINT | 矩形子区域左上角在输入图像中的x坐标 |
| nLocY1 | DINT | 矩形子区域左上角在输入图像中的y坐标 |
| nLocX2 | DINT | 矩形子区域右下角在输入图像中的x坐标 |
| nLocY2 | DINT | 矩形子区域右下角在输入图像中的y坐标 |
| nDestImage | REFERENCE TO ULINT | 获取的矩形子区域图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgSetSub



在输入图像中修改矩形子区域，通过定义子区域在输入图像中的左上角坐标。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgSetSub: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImageSub : ULINT;

nLocX1 : DINT := 100;

nLocY1 : DINT := 100;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

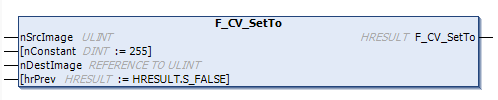
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 要修改的输入图像 |
| nSrcImageSub | ULINT | 待修改的矩形子区域 |
| nLocX1 | DINT | 矩形子区域左上角在输入图像中的位置的x坐标 |
| nLocY1 | DINT | 矩形子区域左上角在输入图像中的位置的y坐标 |
| nDestImage | REFERENCE TO ULINT | 修改后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_SetTo



填充常数图像，将输入图像中的所有像素设置为指定的像素值。

**Syntax**

Definition:

FUNCTION F\_CV\_SetTo: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nConstant : DINT := 255;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

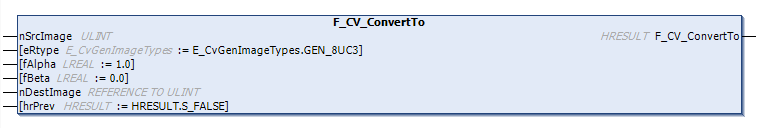
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nConstant | DINT | 指定的要转换的像素值 |
| nDestImage | REFERENCE TO ULINT | 转换像素值后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ConvertTo



将输入图像的类型转换为指定的图像类型。

**Syntax**

Definition:

FUNCTION F\_CV\_ConvertTo: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eRtype : E\_CvGenImageTypes := E\_CvGenImageTypes.GEN\_8UC3;

fAlpha : LREAL := 1.0;

fBeta : LREAL := 0.0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| eRtype | E\_CvGenImageTypes | 指定要转换的图像类型，枚举列表如下 |
| fAlpha | LREAL | 比例因子 |
| fBeta | LREAL | 添加到缩放值 |
| nDestImage | REFERENCE TO ULINT | 输出类型被转换后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

图像类型

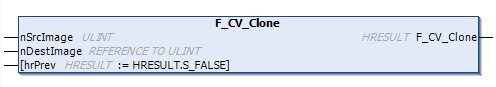
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| GEN\_8UC1 | GEN\_8SC1 | GEN\_16UC1 | GEN\_16SC1 | GEN\_32SC1 | GEN\_32FC1 | GEN\_64FC1 | GEN\_16FC1 |
| GEN\_8UC2 | GEN\_8SC2 | GEN\_16UC2 | GEN\_16SC2 | GEN\_32SC2 | GEN\_32FC2 | GEN\_64FC2 | GEN\_16FC2 |
| GEN\_8UC3 | GEN\_8SC3 | GEN\_16UC3 | GEN\_16SC3 | GEN\_32SC3 | GEN\_32FC3 | GEN\_64FC3 | GEN\_16FC3 |
| GEN\_8UC4 | GEN\_8SC4 | GEN\_16UC4 | GEN\_16SC4 | GEN\_32SC4 | GEN\_32FC4 | GEN\_64FC4 | GEN\_16FC4 |
| GEN\_8UC5 | GEN\_8SC5 | GEN\_16UC5 | GEN\_16SC5 | GEN\_32SC5 | GEN\_32FC5 | GEN\_64FC5 | GEN\_16FC5 |
| GEN\_8UC6 | GEN\_8SC6 | GEN\_16UC6 | GEN\_16SC6 | GEN\_32SC6 | GEN\_32FC6 | GEN\_64FC6 | GEN\_16FC6 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Clone



复制，将输入图像深度拷贝到新的图像。

**Syntax**

Definition:

FUNCTION F\_CV\_Clone: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

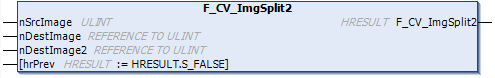
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| nDestImage | REFERENCE TO ULINT | 复制后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgSplit2



双通道分解，输入图像为一张双通道图像，将图片按照通道分解，分解为两个独立的单通道图像。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgSplit2: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

nDestImage2 : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

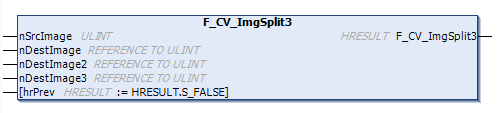
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的双通道图像，图像通道必须是2 |
| nDestImage | REFERENCE TO ULINT | 分解后的第一通道的图像 |
| nDestImage2 | REFERENCE TO ULINT | 分解后的第二通道的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgSplit3



三通道分解，输入图像为一张三通道图像，将图片按照通道分解，分解为三个独立的单通道图像。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgSplit3: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

nDestImage2 : REFERENCE TO ULINT;

nDestImage3 : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

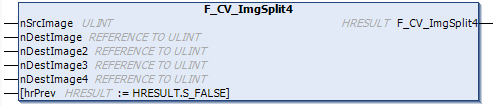
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的三通道图像，图像通道必须是3 |
| nDestImage | REFERENCE TO ULINT | 分解后的第一通道的图像 |
| nDestImage2 | REFERENCE TO ULINT | 分解后的第二通道的图像 |
| nDestImage3 | REFERENCE TO ULINT | 分解后的第三通道的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgSplit4



四通道分解，输入图像为一张四通道图像，将图片按照通道分解，分解为四个独立的单通道图像。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgSplit4: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

nDestImage2 : REFERENCE TO ULINT;

nDestImage3 : REFERENCE TO ULINT;

nDestImage4 : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

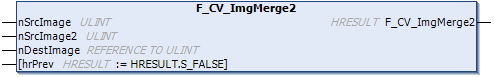
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的四通道图像，图像通道必须是4 |
| nDestImage | REFERENCE TO ULINT | 分解后的第一通道的图像 |
| nDestImage2 | REFERENCE TO ULINT | 分解后的第二通道的图像 |
| nDestImage3 | REFERENCE TO ULINT | 分解后的第三通道的图像 |
| nDestImage4 | REFERENCE TO ULINT | 分解后的第四通道的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgMerge2



双通道合并，将两个输入的单通道图像，合并为一张双通道的图像。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgMerge2: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

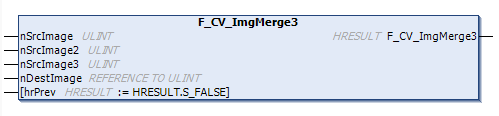
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的第一个单通道图像，必须是单通道 |
| nSrcImage2 | ULINT | 输入的第二个单通道图像，必须是单通道 |
| nDestImage | REFERENCE TO ULINT | 合并后的双通道图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgMerge3



三通道合并，将三个输入的单通道图像，合并为一张三通道的图像。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgMerge3: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nSrcImage3 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

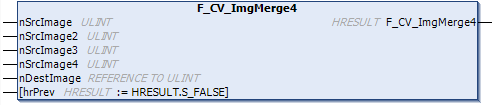
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的第一个单通道图像，必须是单通道 |
| nSrcImage2 | ULINT | 输入的第二个单通道图像，必须是单通道 |
| nSrcImage3 | ULINT | 输入的第三个单通道图像，必须是单通道 |
| nDestImage | REFERENCE TO ULINT | 合并后的三通道图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgMerge4



四通道合并，将四个输入的单通道图像，合并为一张四通道的图像。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgMerge4: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nSrcImage3 : ULINT;

nSrcImage4 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

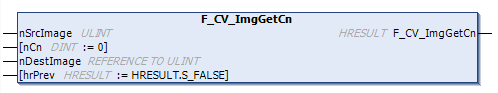
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的第一个单通道图像，必须是单通道 |
| nSrcImage2 | ULINT | 输入的第二个单通道图像，必须是单通道 |
| nSrcImage3 | ULINT | 输入的第三个单通道图像，必须是单通道 |
| nSrcImage4 | ULINT | 输入的第四个单通道图像，必须是单通道 |
| nDestImage | REFERENCE TO ULINT | 合并后的四通道图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImgGetCn



获取指定通道的图像。指定一个通道号，从输入的图像中获取指定通道的图像，获得一张单通道图。

**Syntax**

Definition:

FUNCTION F\_CV\_ImgGetCn: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nCn : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

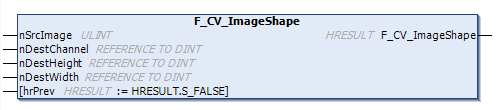
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像 |
| nCn | DINT | 通道的编号，大于等于0，小于输入图像本身的通道数 |
| nDestImage | REFERENCE TO ULINT | 获取的单通道图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImageShape



获取图像大小，给出输入图像的通道数、高、宽。

**Syntax**

Definition:

FUNCTION F\_CV\_ImageShape: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestChannel : REFERENCE TO DINT;

nDestHeight : REFERENCE TO DINT;

nDestWidth : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

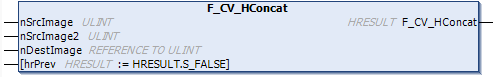
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片 |
| nDestChannel | REFERENCE TO DINT | 输出图片的通道数 |
| nDestHeight | REFERENCE TO DINT | 输出图片的高 |
| nDestWidth | REFERENCE TO DINT | 输出图片的宽 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_HConcat



图像组合-水平，输入两张相同类型、相同通道数的图片。两张图片的高必须相同，将他们水平排列，组合成一张图片，组合后的图片的宽为两张输入图片的宽的和。

**Syntax**

Definition:

FUNCTION F\_CV\_HConcat: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

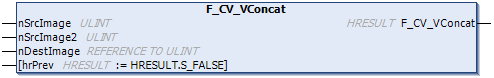
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 用来水平排列的第一张图片 |
| nSrcImage2 | ULINT | 用来水平排列的第二张图片 |
| nDestImage | REFERENCE TO ULINT | 输出合并后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_VConcat



图像组合-竖直，输入两张相同类型、相同通道数的图片。两张图片的宽必须相同，将他们竖直排列，组合成一张图片，组合后的图片的高为两张输入图片的高的和。

**Syntax**

Definition:

FUNCTION F\_CV\_VConcat: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

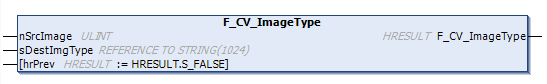
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 用来竖直排列的第一张图片 |
| nSrcImage2 | ULINT | 用来竖直排列的第二张图片 |
| nDestImage | REFERENCE TO ULINT | 输出合并后的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ImageType



获取输入图片的类型，以字符串形式输出。

**Syntax**

Definition:

FUNCTION F\_CV\_ImageType: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

sDestImgType : REFERENCE TO STRING(1024);

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像 |
| sDestImgType | REFERENCE TO STRING | 获得的图像类型名称，类型列别如下 |
| hrPrev | HRESULT | 算子的初始执行状况 |

图像类型包括

|  |  |  |  |
| --- | --- | --- | --- |
| 8UC1 | 16UC1 | 32SC1 | 64FC1 |
| 8UC2 | 16UC2 | 32SC2 | 64FC2 |
| 8UC3 | 16UC3 | 32SC3 | 64FC3 |
| 8UC4 | 16UC4 | 32SC4 | 64FC4 |
| 8UC5 | 16UC5 | 32SC5 | 64FC5 |
| 8UC6 | 16UC6 | 32SC6 | 64FC6 |
| 8SC1 | 16SC1 | 32FC1 | 16FC1 |
| 8SC2 | 16SC2 | 32FC2 | 16FC2 |
| 8SC3 | 16SC3 | 32FC3 | 16FC3 |
| 8SC4 | 16SC4 | 32FC4 | 16FC4 |
| 8SC5 | 16SC5 | 32FC5 | 16FC5 |
| 8SC6 | 16SC6 | 32FC6 | 16FC6 |

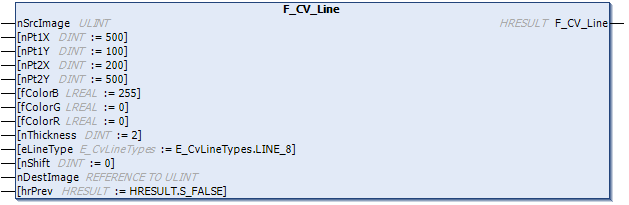
**Return value**

HRESULT

**Further information**

### 图像绘制

#### F\_CV\_Line



输入图像并绘制连接两点的线段。线会被图像边界剪切。对于非抗锯齿线，使用8连通或4连通的Bresenham算法；对于抗锯齿线，使用高斯滤波绘制。粗线以圆边绘制。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Line: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nPt1X : DINT := 500;

nPt1Y : DINT := 100;

nPt2X : DINT := 200;

nPt2Y : DINT := 500;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

nShift : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

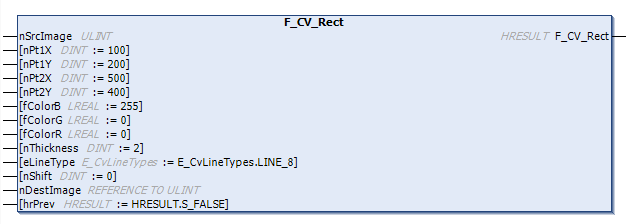
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nPt1X | DINT | 线段的第一点，顶点x坐标 |
| nPt1Y | DINT | 线段的第一点，顶点y坐标， |
| nPt2X | DINT | 线段的第二个点，对应顶点的x坐标 |
| nPt2Y | DINT | 线段的第二个点，对应顶点的y坐标 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nThickness | DINT | 线条粗细 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| nShift | DINT | 把中心点坐标以及线的长度按比例缩小 假设原来的数字为D，变换后的数字为d， 则具体公式是d = D/(2^shift) |
| nDestImage | REFERENCE  TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Rect



绘制矩形，绘制一个矩形轮廓或一个填充矩形，定义矩形两个相对的角。

**Syntax**

Definition:

FUNCTION F\_CV\_Rect: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nPt1X : DINT := 100;

nPt1Y : DINT := 200;

nPt2X : DINT := 500;

nPt2Y : DINT := 400;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

nShift : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

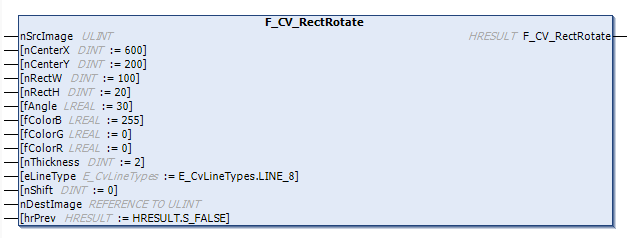
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nPt1X | DINT | 矩形顶点x坐标 |
| nPt1Y | DINT | 矩形顶点y坐标 |
| nPt2X | DINT | 矩形对应顶点x坐标 |
| nPt2Y | DINT | 矩形对应顶点y坐标 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nThickness | DINT | 线条粗细，如果为正数；如通为负数，填充轮廓 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| nShift | DINT | 把中心点坐标以及线的长度按比例缩小 假设原来的数字为D，变换后的数字为d， 则具体公式是d = D/(2^shift) |
| nDestImage | REFERENCE  TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_RectRotate



绘制旋转矩形，定义矩形的中心点与宽高，通过定义旋转角度，使得矩形围绕中心点顺时针旋转。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_RectRotate: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nCenterX : DINT := 600;

nCenterY : DINT := 200;

nRectW : DINT := 100;

nRectH : DINT := 20;

fAngle : LREAL := 30;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

nShift : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

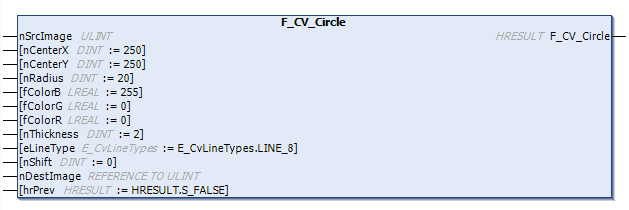
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nCenterX | DINT | 旋转矩形中心x坐标 |
| nCenterY | DINT | 旋转矩形中心y坐标 |
| nRectW | DINT | 旋转矩形宽 |
| nRectH | DINT | 旋转矩形长 |
| fAngle | LREAL | 矩形旋转角度 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nThickness | DINT | 线条粗细 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| nShift | DINT | 把中心点坐标以及线的长度按比例缩小， 假设原来的数字为D，变换后的数字为d，  则具体公式是d = D/(2^shift) |
| nDestImage | REFERENCE TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Circle



绘制圆形，通过定义圆的圆心与半径。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Circle: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nCenterX : DINT := 250;

nCenterY : DINT := 250;

nRadius : DINT := 20;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

nShift : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

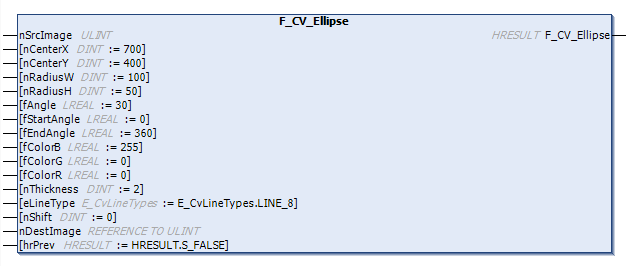
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nCenterX | DINT | 中心点x坐标 |
| nCenterY | DINT | 中心点y坐标 |
| nRadius | DINT | 半径 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nThickness | DINT | 线条粗细，如果为正数；如通为负数，填充轮廓 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| nShift | DINT | 把中心点坐标以及线的长度按比例缩小， 假设原来的数字为D，变换后的数字为d，  则具体公式是d = D/(2^shift) |
| nDestImage | REFERENCE TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Ellipse



绘制椭圆、圆弧、或填充一个椭圆扇区，椭圆可以通过圆心与两个方向的半径来控制，圆弧可以通过定义起始角度与结束角度来控制。如果绘制一个完整的椭圆，而不是圆弧，则通过设置起始角度为0和结束角度为360。如果起始角度大于结束角度，它们将被交换。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Ellipse: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nCenterX : DINT := 700;

nCenterY : DINT := 400;

nRadiusW : DINT := 100;

nRadiusH : DINT := 50;

fAngle : LREAL := 30;

fStartAngle : LREAL := 0;

fEndAngle : LREAL := 360;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

nShift : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

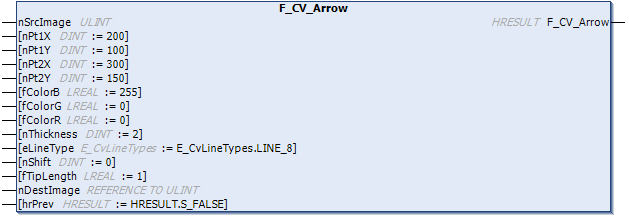
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nCenterX | DINT | 中心点x坐标 |
| nCenterY | DINT | 中心点y坐标 |
| nRadiusW | DINT | 半径沿x方向，椭圆主轴大小的一半。 |
| nRadiusH | DINT | 半径沿y方向，椭圆主轴大小的一半。 |
| fAngle | LREAL | 椭圆旋转角度（以度为单位） |
| fStartAngle | LREAL | 椭圆弧的起始角（度） |
| fEndAngle | LREAL | 椭圆弧的终止角（度） |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nThickness | DINT | 线条粗细，如果为正数；如通为负数，填充轮廓 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| nShift | DINT | 把中心点坐标以及线的长度按比例缩小， 假设原来的数字为D，变换后的数字为d，  则具体公式是d = D/(2^shift) |
| nDestImage | REFERENCE TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Arrow



箭头，绘制从第一个点指向第二个点的箭头线段。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Arrow: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nPt1X : DINT := 200;

nPt1Y : DINT := 100;

nPt2X : DINT := 300;

nPt2Y : DINT := 150;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

nShift : DINT := 0;

fTipLength : LREAL := 1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

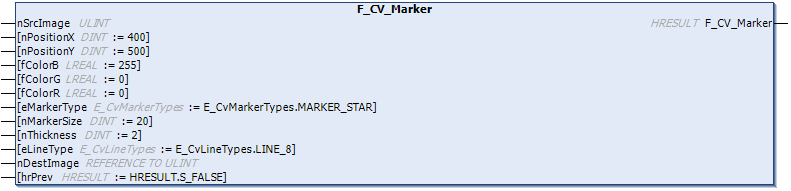
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nPt1X | DINT | 箭头的起点，x坐标 |
| nPt1Y | DINT | 箭头的起点，y坐标 |
| nPt2X | DINT | 箭头指向的点，x坐标 |
| nPt2Y | DINT | 箭头指向的点，y坐标 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nThickness | DINT | 线条粗细 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| nShift | DINT | 把中心点坐标以及线的长度按比例缩小  假设原来的数字为D，变换后的数字为d 则具体公式是d = D/(2^shift) |
| fTipLength | LREAL | 箭头尖端的长度，相对于箭头长度 |
| nDestImage | REFERENCE  TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Marker



标记，在图像中的预定义位置绘制标记，多种标记类型可以被选择。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Marker: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nPositionX : DINT := 400;

nPositionY : DINT := 500;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

eMarkerType : E\_CvMarkerTypes := E\_CvMarkerTypes.MARKER\_STAR;

nMarkerSize : DINT := 20;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

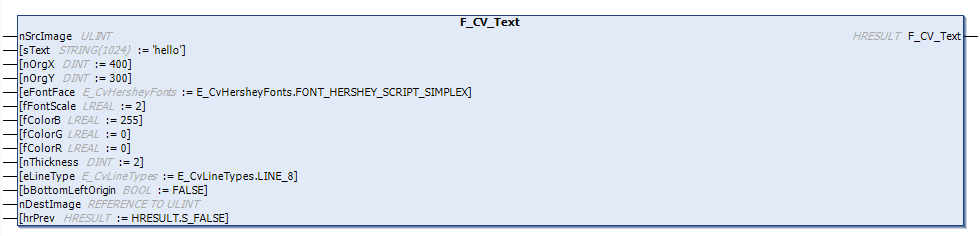
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nPositionX | DINT | 十字光标所在的点，x坐标 |
| nPositionY | DINT | 十字光标所在的点，y坐标 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| eMarkerType | E\_CvMarkerTypes | 要使用的标记的特定类型   1. MARKER\_CROSS: 十字； 2. MARKER\_DIAMOND: 钻石； 3. MARKER\_SQUARE: 正方形； 4. MARKER\_STAR: 星形； 5. MARKER\_TILTED\_CROSS: 叉叉； 6. MARKER\_TRIANGLE\_DOWN: 向下箭头； 7. MARKER\_TRIANGLE\_UP: 向上箭头； |
| nMarkerSize | DINT | 标记轴的长度[默认值=20像素] |
| nThickness | DINT | 线条粗细 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| nDestImage | REFERENCE  TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Text



绘制文字，无法使用的字体符号将被问号替换，目前只支持英文，通过移动文字框左下角的坐标来绘制到指定位置，文字的字体与大小也可以被定义。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Text: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

sText : STRING(1024) := 'hello';

nOrgX : DINT := 400;

nOrgY : DINT := 300;

eFontFace : E\_CvHersheyFonts :=

E\_CvHersheyFonts.FONT\_HERSHEY\_SCRIPT\_SIMPLEX;

fFontScale : LREAL := 2;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nThickness : DINT := 2;

eLineType : E\_CvLineTypes := E\_CvLineTypes.LINE\_8;

bBottomLeftOrigin : BOOL := FALSE;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

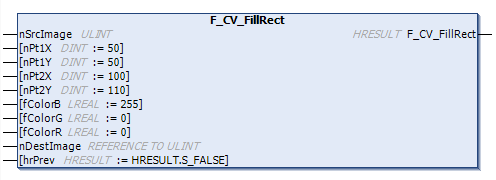
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| sText | STRING | 要绘制的文本字符串 |
| nOrgX | DINT | 图像中文本字符串的左下角x坐标 |
| nOrgY | DINT | 图像中文本字符串的左下角y坐标 |
| eFontFace | E\_CvHersheyFonts | 字体类型   1. FONT\_HERSHEY\_SIMPLEX: 正常大小无衬线字体； 2. FONT\_HERSHEY\_PLAIN: 小号无衬线字体； 3. FONT\_HERSHEY\_DUPLEX: 正常大小无衬线字体，比FONT\_HERSHEY\_SIMPLEX更复杂； 4. FONT\_HERSHEY\_COMPLEX: 正常大小有衬线字体； 5. FONT\_HERSHEY\_TRIPLEX: 正常大小有衬线字体，比FONT\_HERSHEY\_COMPLEX更复杂； 6. FONT\_HERSHEY\_COMPLEX\_SMALL: FONT\_HERSHEY\_COMPLEX的小译本； 7. FONT\_HERSHEY\_SCRIPT\_SIMPLEX: 手写风格字体； 8. FONT\_HERSHEY\_SCRIPT\_COMPLEX: 手写风格字体，比FONT\_HERSHEY\_SCRIPT\_SIMPLEX更复杂，这些参数和FONT\_ITALIC同时使用就会得到相应的斜体字； |
| fFontScale | LREAL | 字体比例因子，乘以特定于字体的基本大小 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nThickness | DINT | 线条粗细 |
| eLineType | E\_CvLineTypes | 线的类型   1. FILLED: 内部填充（实心图形）； 2. LINE\_4: 4连通钜齿线； 3. LINE\_8: 8连通钜齿线； 4. LINE\_AA: 抗钜齿线； |
| bBottomLeftOrigin | BOOL | 如果为true，则图像数据原点位于左下角。否则，它在左上角 |
| nDestImage | REFERENCE  TO ULINT | 绘制后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_FillRect



在输入的图片中，选择一个方形区域涂色，将指定区域涂成特定颜色。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_FillRect: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nPt1X : DINT := 50;

nPt1Y : DINT := 50;

nPt2X : DINT := 100;

nPt2Y : DINT := 110;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

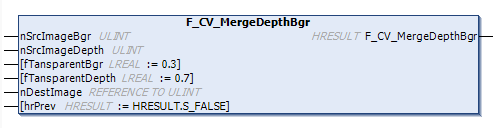
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，即要绘制的源图 |
| nPt1X | DINT | 矩形的左上角顶点x坐标 |
| nPt1Y | DINT | 矩形的左上角顶点y坐标 |
| nPt2X | DINT | 与左上角相对的矩形的顶点x坐标 |
| nPt2Y | DINT | 与左上角相对的矩形的顶点y坐标 |
| fColorB | LREAL | 要涂的指定颜色，三原色中的蓝色，若是黑白只选蓝色 |
| fColorG | LREAL | 要涂的指定颜色，三原色中的绿色，若是黑白只选蓝色 |
| fColorR | LREAL | 要涂的指定颜色，三原色中的红色，若是黑白只选蓝色 |
| nDestImage | REFERENCE  TO ULINT | 区域被涂色后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_MergeDepthBgr



将输入的深度图与RGB彩色图重合对齐，通过控制两者的透明度，生成一张合并图，从而看清图像中物体的深度信息。

**Syntax**

Definition:

FUNCTION F\_CV\_MergeDepthBgr: HRESULT

VAR\_INPUT

nSrcImageBgr : ULINT;

nSrcImageDepth : ULINT;

fTansparentBgr : LREAL := 0.3;

fTansparentDepth : LREAL := 0.7;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

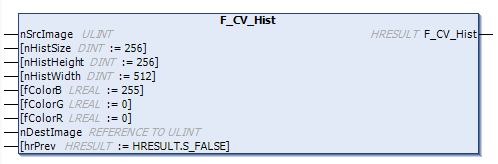
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImageBgr | ULINT | 输入彩色图，为BGR格式 |
| nSrcImageDepth | ULINT | 输入深度图，数据类型为16U |
| fTansparentBgr | LREAL | 彩色图透明度 |
| fTansparentDepth | LREAL | 深度图透明度 |
| nDestImage | REFERENCE TO ULINT | 合并后的图片 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Hist



绘制直方图，通过函数F\_CV\_CalcHist获得图像的直方图信息，再通过此函数绘制直方图来展示。

**Syntax**

Definition:

FUNCTION F\_CV\_Hist: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nHistSize : DINT := 256;

nHistHeight : DINT := 256;

nHistWidth : DINT := 512;

fColorB : LREAL := 255;

fColorG : LREAL := 0;

fColorR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入函数F\_CV\_CalcHist计算的直方图数据 |
| nHistSize | DINT | 直方图中每个维度需要分成多少个区间（如果把直方图看作一个一个竖条的话，就是竖条的个数） |
| nHistHeight | DINT | 直方图高 |
| nHistWidth | DINT | 直方图宽，图片宽度必须大于高度，否则返回白图 |
| fColorB | LREAL | 线条颜色，三原色中的蓝色 |
| fColorG | LREAL | 线条颜色，三原色中的绿色 |
| fColorR | LREAL | 线条颜色，三原色中的红色 |
| nDestImage | REFERENCE  TO ULINT | 输出直方图展示函数F\_CV\_CalcHist计算的直方图数据 |
| hrPrev | HRESULT | 算子的初始执行状况 |

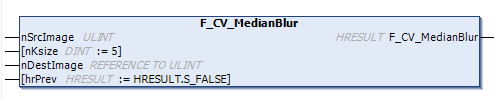
**Return value**

HRESULT

**Further information**

### 图像处理

#### F\_CV\_MedianBlur



中值滤波，使用中值滤波器模糊图像，在内部使用BORDER\_REPLICATE来处理边界像素。该函数使用nKsize×nKsize孔径的中值滤波器来平滑图像。多通道图像的每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_MedianBlur: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nKsize : DINT := 5;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

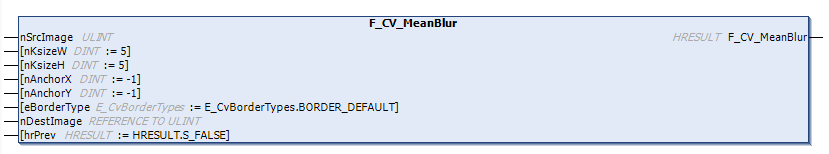
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入1、3或4通道图像；当nKsize为3或5时，图像深度应为8U、16U或32F；对于较大的孔径尺寸，只能为8U |
| nKsize | DINT | 孔径线性尺寸；它必须是奇数并且大于1，例如：3，5，7... |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_MeanBlur



均值滤波，使用标准化的长方体过滤器使图像变模糊，平滑内核：  
K = 1/(nKsizeW\*nKsizeH )

**Syntax**

Definition:

FUNCTION F\_CV\_MeanBlur: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

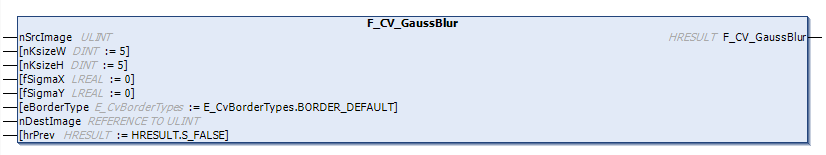
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像；它可以有任意数量的通道，这些通道是独立处理的，但深度应该是8U、16U、1 6S、32F或64F |
| nKsizeW | DINT | 模糊内核宽，必须是正数和奇数 |
| nKsizeH | DINT | 模糊内核高，必须是正数和奇数 |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_GaussBlur



高斯滤波，使用高斯滤波器使图像变模糊，通过调用高斯内核。高斯模糊是一种线性滤波技术，通过对图像中每个像素及其邻域像素进行加权平均来实现模糊效果。加权平均的权重由一个高斯函数决定，因此称为高斯模糊。高斯模糊可以有效地减少图像中的噪声和细节，使图像变得平滑。

**Syntax**

Definition:

FUNCTION F\_CV\_GaussBlur: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

fSigmaX : LREAL := 0;

fSigmaY : LREAL := 0;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

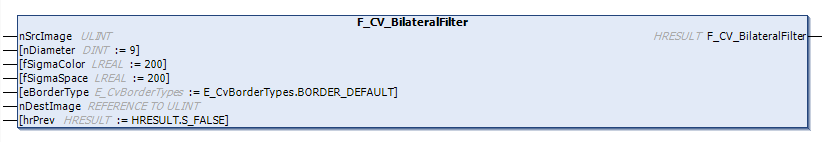
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像；图像可以具有任意数量的通道，这些通道是独立处理的，但深度应该是8U、16U、1 6S、32F或64F。 |
| nKsizeW | DINT | 高斯内核宽，可以与高不同，但它们都必须是正数和奇数 |
| nKsizeH | DINT | 高斯内核高，可以与宽不同，但它们都必须是正数和奇数 或者，它们可以是零，然后根据sigma计算。 |
| fSigmaX | LREAL | X方向上的高斯核标准差 |
| fSigmaY | LREAL | Y方向上的高斯核标准差 如果fSigmaY为零，则设置为等于fSigmaX；如果两个sigma都为零，分别根据nKsizeW和nKsizeH计算；为了完全控制结果，而不管将来可能对所有这些语义进行修改，建议指定所有nKsizeW、nKsizeH、fSigmaX和fSigmaY。 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BilateralFilter



双边滤波，将双向过滤器应用于图像，双边滤波器可以很好地减少不必要的噪声，同时保持边缘相当清晰。然而，与大多数过滤器相比，它非常慢。对于 fSigmaColor 和 fSigmaSpace 的取值，如果它们很小，如<10，滤镜不会产生太大效果；而如果它们很大，如>150，它们会产生非常强烈的效果，使图像看起来“卡通”；对于过滤器大小 nDiameter 的取值，大型过滤器，如>5则非常慢，因此建议对实时应用程序取值5，对需要大量噪声过滤的离线应用程序则可以取值9。

**Syntax**

Definition:

FUNCTION F\_CV\_BilateralFilter: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDiameter : DINT := 9;

fSigmaColor : LREAL := 200;

fSigmaSpace : LREAL := 200;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

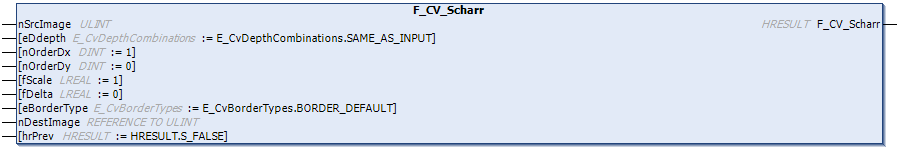
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源8位或浮点、1通道或3通道图像 |
| nDiameter | DINT | 过滤过程中使用的每个像素邻域的直径。如果它是非正的，它是从fSigmaSpace计算出来的。 |
| fSigmaColor | LREAL | 在颜色空间中过滤 sigma 参数。该参数的值越大，意味着像素邻域内更远的颜色（请参见fSigmaSpace）将混合在一起，从而产生更大的半等色区域。 |
| fSigmaSpace | LREAL | 在坐标空间中sigma参数。该参数的值越大，意味着只要颜色足够接近，越远的像素就会相互影响。当nDiameter>0时，它指定邻域大小，而不考虑fSigmaSpace。否则，nDiameter与fSigmaSpace成比例。 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Scharr



SCHARR滤波，使用Scharr算子分别计算x方向或y方向的图像差分，由两个卷积核组成，且两个卷积核均是不可分离的。图像与水平方向上的卷积结果反响的是垂直方向上的边缘强度，垂直方向上的卷积结果反映的是水平方向上的边缘强度。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Scharr: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eDdepth : E\_CvDepthCombinations := E\_CvDepthCombinations.SAME\_AS\_INPUT;

nOrderDx : DINT := 1;

nOrderDy : DINT := 0;

fScale : LREAL := 1;

fDelta : LREAL := 0;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

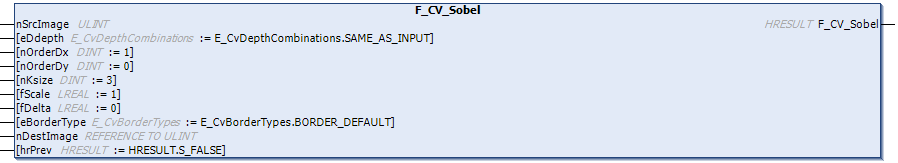
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像 |
| eDdepth | E\_CvDepthCombinations | 输出图像深度   1. 输入深度：8U，输出深度：-1/16S/32F/64F； 2. 输入深度：16U/16S，输出深度：-1/32F/64F； 3. 输入深度：32F，输出深度：-1/32F/64F； 4. 输入深度：64F，输出深度：-1/64F；   注意：当eDdepth=-1时，输出图像将具有与源相同的深度。 |
| nOrderDx | DINT | 导数x的阶数 |
| nOrderDy | DINT | 导数y的阶数 |
| fScale | LREAL | 计算的导数值的可选比例因子；默认情况下，不应用缩放 |
| fDelta | LREAL | 可选的增量值，在将结果存储到结果之前先添加到结果中。 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| nDestImage | REFERENCE TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Sobel



Sobel边缘检测，使用扩展的Sobel算子计算一阶、二阶、三阶或混合图像导数。当 nKsize=1 时，只能用于第一阶或第二阶的x或y导数，使用3x1或1x3核来计算导数（即不进行高斯平滑）；其他情况使用 nKsize\*nKsize 可分离核来计算导数。Sobel算子是一个用于边缘检测的离散微分算子，它结合了高斯平滑和微分求导，用来计算图像灰度函数的近似梯度。在图像的任何一点使用此算子，都会得倒对应的梯度矢量或其法向量。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Sobel: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eDdepth : E\_CvDepthCombinations := E\_CvDepthCombinations.SAME\_AS\_INPUT;

nOrderDx : DINT := 1;

nOrderDy : DINT := 0;

nKsize : DINT := 3;

fScale : LREAL := 1;

fDelta : LREAL := 0;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

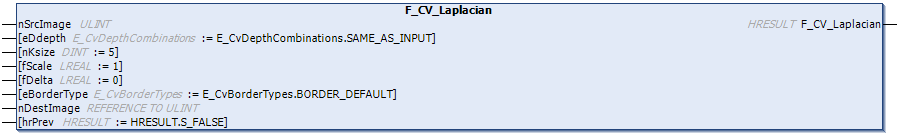
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像。 |
| eDdepth | E\_CvDepthCombinations | 输出图像深度   1. 输入深度：8U，输出深度：-1/16S/32F/64F； 2. 输入深度：16U/16S，输出深度：-1/32F/64F； 3. 输入深度：32F，输出深度：-1/32F/64F； 4. 输入深度：64F，输出深度：-1/64F；   注意：当eDdepth=-1时，输出图像将具有与源相同的深度。 |
| nOrderDx | DINT | 导数x的阶数。 |
| nOrderDy | DINT | 导数y的阶数 |
| nKsize | DINT | 扩展Sobel内核的大小；它必须是1、3、5或7 |
| fScale | LREAL | 计算的导数值的可选比例因子；默认情况下，不应用缩放 |
| fDelta | LREAL | 可选的增量值，在将结果存储到结果之前先添加到结果中。 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| nDestImage | REFERENCE TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Laplacian



Laplacian边缘检测，计算图像的拉普拉斯算子。当nKsize>1时，Sobel算子用来计算二阶的x和y导数，再相加来计算拉普拉斯算子，当 nKsize=1 时，拉普拉斯算子是通过默认的3×3孔径对图像进行滤波来计算。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Laplacian: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eDdepth : E\_CvDepthCombinations := E\_CvDepthCombinations.SAME\_AS\_INPUT;

nKsize : DINT := 5;

fScale : LREAL := 1;

fDelta : LREAL := 0;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

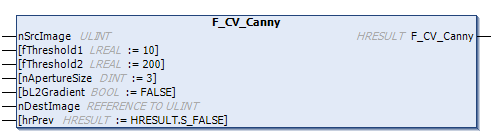
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像。 |
| eDdepth | E\_CvDepthCombinations | 输出图像深度   1. 输入深度：8U，输出深度：-1/16S/32F/64F； 2. 输入深度：16U/16S，输出深度：-1/32F/64F； 3. 输入深度：32F，输出深度：-1/32F/64F； 4. 输入深度：64F，输出深度：-1/64F；   注意：当eDdepth=-1时，输出图像将具有与源相同的深度。 |
| nKsize | DINT | 用于计算二阶导数滤波器的孔径大小。大小必须是正数和奇数。 |
| fScale | LREAL | 计算的拉普拉斯值的可选比例因子。默认情况下，不应用缩放。 |
| fDelta | LREAL | 可选的增量值，在将结果存储到结果之前先添加到结果中。 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| nDestImage | REFERENCE TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Canny



CANNY边缘检测，使用canny算法查找图像中的边缘，该函数在输入图像中查找边缘，并使用Canny算法在输出映射边缘中标记它们。阈值1和阈值2之间的最小值用于边缘链接。最大值用于查找强边的初始段。

**Syntax**

Definition:

FUNCTION F\_CV\_Canny: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fThreshold1 : LREAL := 10;

fThreshold2 : LREAL := 200;

nApertureSize : DINT := 3;

bL2Gradient : BOOL := FALSE;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

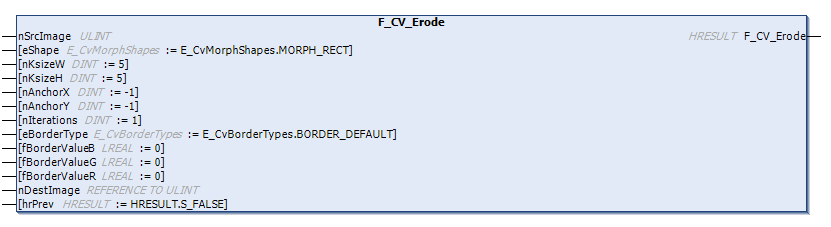
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 8位单通道输入图像 |
| fThreshold1 | LREAL | 滞后过程的第一阈值 |
| fThreshold2 | LREAL | 滞后过程的第二阈值 |
| nApertureSize | DINT | 表示算子的孔径的大小，默认值时3 |
| bL2Gradient | BOOL | 计算图像梯度复制的标识，默认false；如果为 True，则使用更精确的 L2 范数进行计算（即两个方向的导数的平方和再开方）；否则使用 L1 范数（直接将两个方向导数的绝对值相加） |
| nDestImage | REFERENCE  TO ULINT | 输出边缘图；单通道8位图像，其具有与图像相同的大小 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Erode



腐蚀，使用特定的结构元素来侵蚀图像,该结构元素的大小和形状决定了像素邻域的形状。在该邻域上取最小值，通过以公式: dst(x, y) = min src(x+x', y+y')。侵蚀可以施加几次（迭代）。对于多通道图像，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_Erode: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eShape : E\_CvMorphShapes := E\_CvMorphShapes.MORPH\_RECT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

nIterations : DINT := 1;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

fBorderValueB : LREAL := 0;

fBorderValueG : LREAL := 0;

fBorderValueR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

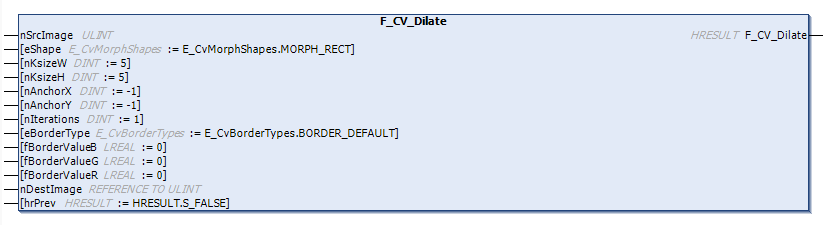
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像；通道的数量可以是任意的，但深度应该是8U、16U、1 6S、32F或64F中的一个。 |
| eShape | E\_CvMorphShapes | 元素形状   1. MORPH\_RECT 函数返回矩形卷积核； 2. MORPH\_CROSS 函数返回十字形卷积核； 3. MORPH\_ELLIPSE 函数返回椭圆形卷积核； |
| nKsizeW | DINT | 用于形态学运算的结构化元素的内核宽，必须是正数和奇数 |
| nKsizeH | DINT | 用于形态学运算的结构化元素的内核高，必须是正数和奇数 |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心 |
| nIterations | DINT | 应用侵蚀的次数 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| fBorderValueB | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的蓝 |
| fBorderValueG | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的绿 |
| fBorderValueR | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的红 |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Dilate



膨胀，使用特定的结构化元素来放大图像，该结构元素的大小和形状决定了像素邻域的形状。在该邻域内取最大值，通过以公式：dst(x, y) = max src(x+x', y+y')。放大可以施加几次（迭代）。对于多通道图像，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_Dilate: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eShape : E\_CvMorphShapes := E\_CvMorphShapes.MORPH\_RECT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

nIterations : DINT := 1;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

fBorderValueB : LREAL := 0;

fBorderValueG : LREAL := 0;

fBorderValueR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

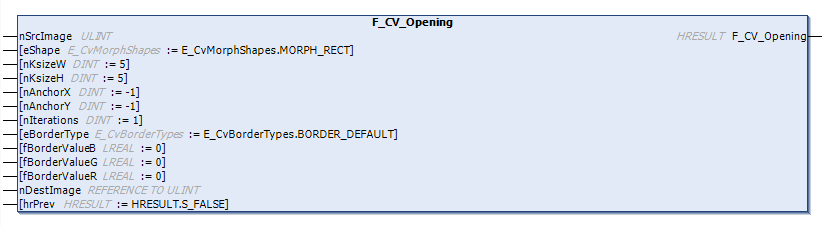
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像；通道的数量可以是任意的，但深度应该是8U、16U、1 6S、32F或64F中的一个。 |
| eShape | E\_CvMorphShapes | 元素形状   1. MORPH\_RECT 函数返回矩形卷积核； 2. MORPH\_CROSS 函数返回十字形卷积核； 3. MORPH\_ELLIPSE 函数返回椭圆形卷积核； |
| nKsizeW | DINT | 用于形态学运算的结构化元素的内核宽，必须是正数和奇数 |
| nKsizeH | DINT | 用于形态学运算的结构化元素的内核高，必须是正数和奇数 |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心 |
| nIterations | DINT | 应用膨胀的次数 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| fBorderValueB | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的蓝 |
| fBorderValueG | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的绿 |
| fBorderValueR | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的红 |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Opening



图像开运算，即对图像先腐蚀，后膨胀。图像被腐蚀后，去除了噪声，但是也压缩了图像；接着对腐蚀过的图像进行膨胀处理，可以去除噪声，并保留原有图像。开运算可以用来消除小物体、在纤细点处分离物体、平滑较大物体的边界的同时并不明显改变其面积。

**Syntax**

Definition:

FUNCTION F\_CV\_Opening: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eShape : E\_CvMorphShapes := E\_CvMorphShapes.MORPH\_RECT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

nIterations : DINT := 1;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

fBorderValueB : LREAL := 0;

fBorderValueG : LREAL := 0;

fBorderValueR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

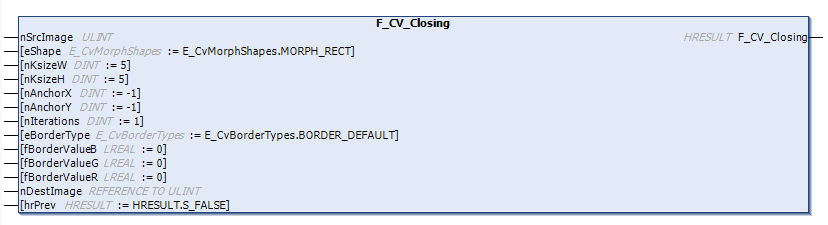
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像；通道的数量可以是任意的，但深度应该是8U、16U、1 6S、32F或64F中的一个。 |
| eShape | E\_CvMorphShapes | 元素形状   1. MORPH\_RECT 函数返回矩形卷积核； 2. MORPH\_CROSS 函数返回十字形卷积核； 3. MORPH\_ELLIPSE 函数返回椭圆形卷积核； |
| nKsizeW | DINT | 用于形态学运算的结构化元素的内核宽，必须是正数和奇数 |
| nKsizeH | DINT | 用于形态学运算的结构化元素的内核高，必须是正数和奇数 |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心 |
| nIterations | DINT | 应用侵蚀和膨胀的次数 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| fBorderValueB | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的蓝 |
| fBorderValueG | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的绿 |
| fBorderValueR | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的红 |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Closing



图像闭运算，即对图像先膨胀，后腐蚀。有助于关闭前景物体内部的小孔，或物体上的小黑点。闭运算能够排除小型黑洞。

**Syntax**

Definition:

FUNCTION F\_CV\_Closing: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eShape : E\_CvMorphShapes := E\_CvMorphShapes.MORPH\_RECT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

nIterations : DINT := 1;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

fBorderValueB : LREAL := 0;

fBorderValueG : LREAL := 0;

fBorderValueR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

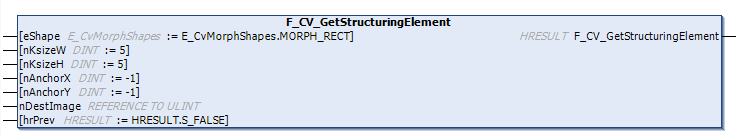
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像；通道的数量可以是任意的，但深度应该是8U、16U、1 6S、32F或64F中的一个。 |
| eShape | E\_CvMorphShapes | 元素形状   1. MORPH\_RECT 函数返回矩形卷积核； 2. MORPH\_CROSS 函数返回十字形卷积核； 3. MORPH\_ELLIPSE 函数返回椭圆形卷积核； |
| nKsizeW | DINT | 用于形态学运算的结构化元素的内核宽，必须是正数和奇数 |
| nKsizeH | DINT | 用于形态学运算的结构化元素的内核高，必须是正数和奇数 |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心 |
| nIterations | DINT | 应用侵蚀和膨胀的次数 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| fBorderValueB | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的蓝 |
| fBorderValueG | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的绿 |
| fBorderValueR | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的红 |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_GetStructuringElement



自定义结构元素，定义内核的大小与形状，创建自定义结构元素。生成的结构学元素被进一步传入自定义形态学函数，用于图像形态学操作，如腐蚀和膨胀等。

**Syntax**

Definition:

FUNCTION F\_CV\_GetStructuringElement: HRESULT

VAR\_INPUT

eShape : E\_CvMorphShapes := E\_CvMorphShapes.MORPH\_RECT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

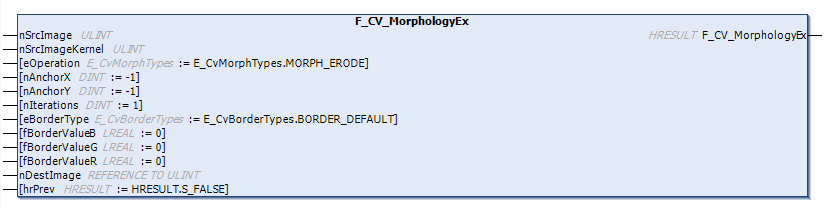
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| eShape | E\_CvMorphShapes | 元素形状   1. MORPH\_RECT 函数返回矩形卷积核； 2. MORPH\_CROSS 函数返回十字形卷积核； 3. MORPH\_ELLIPSE 函数返回椭圆形卷积核； |
| nKsizeW | DINT | 形态学运算的结构化元素的内核宽，必须是正数和奇数 |
| nKsizeH | DINT | 形态学运算的结构化元素的内核高，必须是正数和奇数 |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心。 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心。 注意，只有十字形元件的形状取决于锚固位置；在其他情况下，锚点只是调节形态学运算的结果偏移了多少。 |
| nDestImage | REFERENCE  TO ULINT | 用于形态学运算的指定大小和形状的结构化元素（图像格式） |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_MorphologyEx



自定义形态学，执行高级形态学转换。输入自定义结构元素的形态学内核，选择形态学运算的类型，如，侵蚀和膨胀等，作为基本操作来执行高级形态转换。对于多通道图像，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_MorphologyEx: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImageKernel : ULINT;

eOperation : E\_CvMorphTypes := E\_CvMorphTypes.MORPH\_ERODE;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

nIterations : DINT := 1;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

fBorderValueB : LREAL := 0;

fBorderValueG : LREAL := 0;

fBorderValueR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

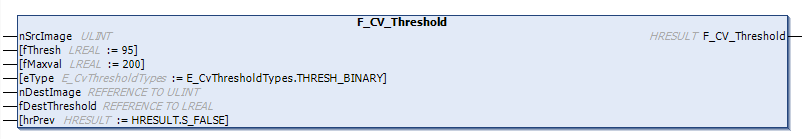
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像。通道的数量可以是任意的。深度应为8U、16U、26S、32F或64F中的一个。 |
| nSrcImageKernel | ULINT | 结构元素，由F\_CV\_GetStructuringElement创建 |
| eOperation | E\_CvMorphTypes | 形态学运算的类型   1. MORPH\_ERODE: 腐蚀； 2. MORPH\_DILATE: 膨胀； 3. MORPH\_OPEN: 开运算； 4. MORPH\_CLOSE: 闭运算； 5. MORPH\_GRADIENT: 形态学梯度； 6. MORPH\_TOPHAT: 顶帽； 7. MORPH\_BLACKHAT: 黑帽； 8. MORPH\_HITMISS: “hit or miss”，只支持8UC1二进制图像； |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心 |
| nIterations | DINT | 应用膨胀、腐蚀等的次数 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| fBorderValueB | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的蓝 |
| fBorderValueG | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的绿 |
| fBorderValueR | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的红 |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Threshold



二值化，将固定级别的阈值应用于每个图像通道，该函数通常用于从灰度图像中获得二进制图像。或用于去除噪声，即滤除值太小或太大的像素。阈值类型THRESH\_OTSU或THRESH\_TRIANGLE。会使用Otsu或Triangle算法来确定最佳的阈值，并使用它来代替输入的阈值。

**Syntax**

Definition:

FUNCTION F\_CV\_Threshold: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fThresh : LREAL := 95;

fMaxval : LREAL := 200;

eType : E\_CvThresholdTypes := E\_CvThresholdTypes.THRESH\_BINARY;

nDestImage : REFERENCE TO ULINT;

fDestThreshold : REFERENCE TO LREAL;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

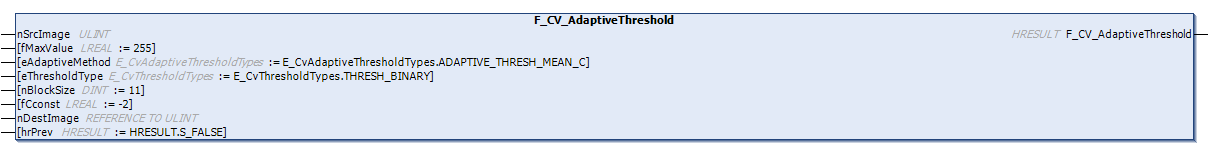
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图（单通道或多通道，8位或32位浮点） |
| fThresh | LREAL | 阈值 |
| fMaxval | LREAL | 用于THRESH\_BINARY和THRESH\_INARY\_INV阈值处理类型的最大值。 |
| eType | E\_CvThresholdTypes | 阈值类型   1. THRESH\_BINARY: 像素值>阈值，像素值=fMaxval；否则像素值=0； 2. THRESH\_BINARY\_INV: 像素值>阈值，像素值=0；否则像素值=fMaxval； 3. THRESH\_TRUNC: 像素值>阈值，像素值=阈值；否则，像素值不变； 4. THRESH\_TOZERO: 像素值>阈值，像素值不变；否则，像素值=0； 5. THRESH\_TOZERO\_INV: 像素值>阈值，像素值=0；否则，像素值不变； 6. THRESH\_OTSU: 仅支持8位单通道图像，使用Otsu算法来确定最佳的阈值； 7. THRESH\_TRIANGLE: 仅支持8位单通道图像，使用Triangle算法来确定最佳的阈值； |
| nDestImage | REFERENCE  TO ULINT | 与输入具有相同大小和类型以及相同通道数的输出图像 |
| fDestThreshold | REFERENCE  TO LREAL | 阈值类型如果使用Otsu或Triangle的方法，则计算出其最佳阈值 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_AdaptiveThreshold



自适应二值化，输入一张单通道图片，将自适应阈值的方法应用于二值化处理图像，相对于固定阈值的二值化处理，自适应二值化对于对比度大的图像有较好效果，自适应阈值处理图像中每一个像素点后，获得的阈值是不同的，该阈值由其领域中图像像素带点加权平均决定。这样做有三点好处：

1. 每个像素位置处的二值化阈值不是固定不变的，而是由其周围邻域像素的分布来决定的；
2. 亮度较高的图像区域的二值化阈值通常会较高，而亮度较低的图像区域的二值化阈值则会相适应地变小；
3. 不同亮度、对比度、纹理的局部图像区域将会拥有相对应的局部二值化阈值；

BORDER\_REPLICATE和BORDER\_ISOLATED的方法被用于处理边界。

**Syntax**

Definition:

FUNCTION F\_CV\_AdaptiveThreshold: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fMaxValue : LREAL := 255;

eAdaptiveMethod : E\_CvAdaptiveThresholdTypes :=   
 E\_CvAdaptiveThresholdTypes.ADAPTIVE\_THRESH\_MEAN\_C;

eThresholdType : E\_CvThresholdTypes := E\_CvThresholdTypes.THRESH\_BINARY;

nBlockSize : DINT := 11;

fCconst : LREAL := -2;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

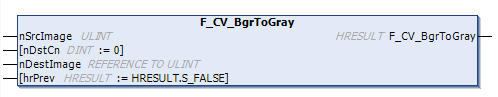
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 8位单通道图像 |
| fMaxValue | LREAL | 指定给满足条件的像素的非零值 |
| eAdaptiveMethod | E\_CvAdaptiveThresholdTypes | 要使用的自适应阈值算法   1. ADAPTIVE\_THRESH\_MEAN\_C: 计算出领域的平均值再减去fCconst； 2. ADAPTIVE\_THRESH\_GAUSSIAN\_C: 算出领域的高斯均值再减去fCconst； |
| eThresholdType | E\_CvThresholdTypes | 阈值类型   1. THRESH\_BINARY: 如果像素值大于阈值，像素值就会被设为参数3;小于等于阈值，设定为0； 2. THRESH\_BINARY\_INV: THRESH\_BINARY的反转，如果像素值大于阈值，像素值为0；小于等于阈值，设定为参数3； |
| nBlockSize | DINT | 用于计算像素阈值的像素邻域的大小：3、5、7，依此类推 |
| fCconst | LREAL | 从平均值或加权平均值中减去的常数。通常情况下，它是正的，但也可能是零或负的。 |
| nDestImage | REFERENCE TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BgrToGray



BGR转灰度，图片由三原色蓝绿红构成的彩色图、三通道，转换为单通道灰度。

**Syntax**

Definition:

FUNCTION F\_CV\_BgrToGray: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDstCn : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

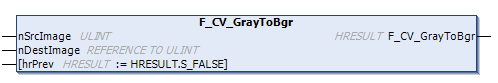
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像，由三原色蓝绿红构成的彩色图、三通道 |
| nDstCn | DINT | 目标图像的通道数，如果参数为0，则通道数将自动推导出，输出的单通道灰度图，与输入大小和类型相同 |
| nDestImage | REFERENCE  TO ULINT | 输出的图片，单通道 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_GrayToBgr



灰度转BGR，将单通道灰度图转换为三通道灰度图，每个通道的像素值一样。

**Syntax**

Definition:

FUNCTION F\_CV\_GrayToBgr: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

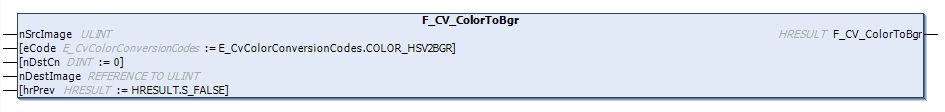
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 单通道灰度图片 |
| nDestImage | REFERENCE TO ULINT | 三通道灰度图, 大小与输入相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ColorToBgr



其它颜色空间转BGR，输入一张图片，图片的颜色空间为选定的颜色空间类型，将其转换为三原色的颜色空间。

**Syntax**

Definition:

FUNCTION F\_CV\_ColorToBgr: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eCode : E\_CvColorConversionCodes :=

E\_CvColorConversionCodes.COLOR\_HSV2BGR;

nDstCn : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

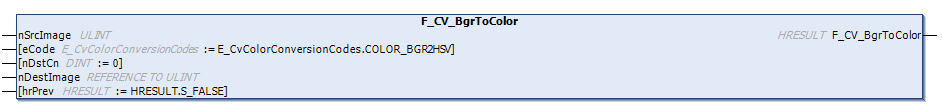
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像，8位无符号、16位无符号（16UC…）或单精度浮点。 |
| eCode | E\_CvColorConversionCodes | 特定以下类型，颜色空间转换代码   1. COLOR\_BGRA2BGR: 输入图片为BGRA类型； 2. COLOR\_XYZ2BGR: 输入图片为XYZ类型； 3. COLOR\_YCrCb2BGR: 输入图片为YCrCb类型； 4. COLOR\_HSV2BGR: 输入图片为HSV类型； 5. COLOR\_Lab2BGR: 输入图片为Lab类型； 6. COLOR\_Luv2BGR: 输入图片为Luv类型； 7. COLOR\_HLS2BGR: 输入图片为HLS类型； 8. COLOR\_YUV2BGR: 输入图片为YUV类型； |
| nDstCn | DINT | 目标图像的通道数，如果参数为0，则通道数将自动推导出。 |
| nDestImage | REFERENCE TO ULINT | 输出图片大小与输入相同的图像。 |
| hrPrev | HRESULT | 算子的初始执行状况。 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BgrToColor



BGR转其它颜色空间，输入一张图片，图片为三原色的颜色空间，将其转换为其它选定的颜色空间。

**Syntax**

Definition:

FUNCTION F\_CV\_BgrToColor: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eCode : E\_CvColorConversionCodes :=

E\_CvColorConversionCodes.COLOR\_BGR2HSV;

nDstCn : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

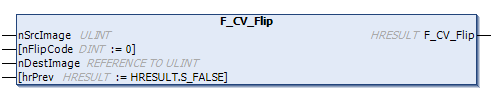
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像，8位无符号、16位无符号（16UC…）或单精度浮点 |
| eCode | E\_CvColorConversionCodes | 特定以下类型，颜色空间转换代码   1. COLOR\_BGR2BGRA: 图片将被转换为BGRA类型； 2. COLOR\_BGR2XYZ: 图片将被转换为XYZ类型； 3. COLOR\_BGR2YCrCb: 图片将被转换为YCrCb类型； 4. COLOR\_BGR2HSV: 图片将被转换为HSV类型； 5. COLOR\_BGR2Lab: 图片将被转换为Lab类型； 6. COLOR\_BGR2Luv: 图片将被转换为Luv类型； 7. COLOR\_BGR2HLS: 图片将被转换为HLS类型； 8. COLOR\_BGR2YUV: 图片将被转换为YUV类型 |
| nDstCn | DINT | 目标图像的通道数，如果参数为0，则通道数将自动推导出 |
| nDestImage | REFERENCE TO ULINT | 输出图片大小与输入相同的图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Flip



镜像，以三种不同的方式之一翻转图片，包括围绕一个轴垂直或水平翻转、以及围绕两个轴翻转，

* 当nFlipCode取0，则垂直翻转图像，以在左上角和左下角图像原点之间切换；
* 当nFlipCode>0，则对图像进行水平翻转，随后进行水平偏移和绝对差分计算，以检查垂直轴对称性；
* 当nFlipCode<0，则同时对图像进行水平和垂直翻转，随后进行偏移和绝对差分计算，以检查中心对称性；
* 当nFlipCode大于或等于0，则反转点阵列的顺序；

坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Flip: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nFlipCode : DINT := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

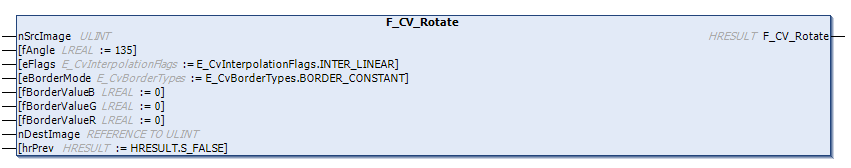
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片 |
| nFlipCode | DINT | 用于指定如何翻转图片，   1. 0：围绕x轴翻转; 2. 正值：如1，围绕y轴翻转; 3. 负值：如-1，围绕x和y两个轴翻转; |
| nDestImage | REFERENCE  TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Rotate



旋转，将输入的图像绕中心点顺时针旋转，旋转至特定的角度。整张旋转后的图像将被展示，边框区域将按照定义的颜色填补。旋转后的图像的大小将按照旋转角度计算。

**Syntax**

Definition:

FUNCTION F\_CV\_Rotate: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fAngle : LREAL := 135;

eFlags : E\_CvInterpolationFlags := E\_CvInterpolationFlags.INTER\_LINEAR;

eBorderMode : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_CONSTANT;

fBorderValueB : LREAL := 0;

fBorderValueG : LREAL := 0;

fBorderValueR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

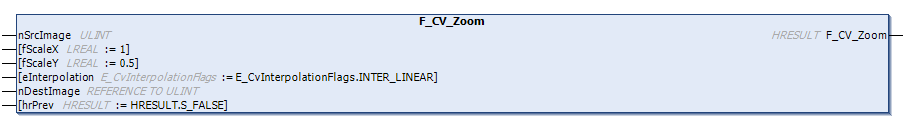
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片。 |
| fAngle | LREAL | 旋转角度（以度为单位），正值表示逆时针旋转（坐标原点假定为左上角）。 |
| eFlags | E\_CvInterpolationFlags | 插值方法   1. INTER\_NEAREST: 最近邻插值； 2. INTER\_LINEAR: 线性插值； 3. INTER\_CUBIC: 三次样条插值； 4. INTER\_AREA: 区域插值； 5. INTER\_LANCZOS4: 8x8像素邻域内的Lanczos插值； 6. INTER\_LINEAR\_EXACT: 精确bit双线性插值； 7. INTER\_NEAREST\_EXACT: 精确bit最近邻插值； 8. INTER\_MAX: 掩码插值； 9. WARP\_FILL\_OUTLIERS: 填充满放大后的目标图像，如果目标图像中对应原图的像素已经溢出，则将其置0； 10. WARP\_INVERSE\_MAP: 反变换； |
| eBorderMode | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如gfedcb|abcdefgh|gfedcba以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| fBorderValueB | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的蓝。 |
| fBorderValueG | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的绿。 |
| fBorderValueR | LREAL | 若边框类型为BORDER\_CONSTANT，边界值中三原色的红。 |
| nDestImage | REFERENCE TO ULINT | 旋转后图像。 |
| hrPrev | HRESULT | 算子的初始执行状况。 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Zoom



比例因子缩放，将输入的图像，按照比例缩放。图像可以沿着x方向或者y方向上缩放，从而改变原图像的长宽比。坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_Zoom: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fScaleX : LREAL := 1;

fScaleY : LREAL := 0.5;

eInterpolation : E\_CvInterpolationFlags := E\_CvInterpolationFlags.INTER\_LINEAR;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

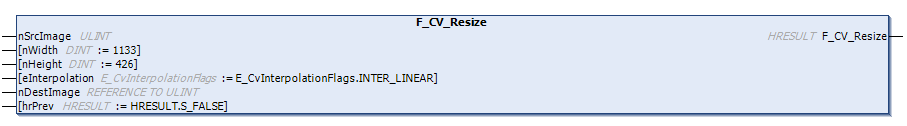
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片 |
| fScaleX | LREAL | 沿水平轴的比例因子 |
| fScaleY | LREAL | 沿垂直轴的比例因子 |
| eInterpolation | E\_CvInterpolationFlags | 插值方法   1. INTER\_NEAREST: 最近邻插值； 2. INTER\_LINEAR: 线性插值； 3. INTER\_CUBIC: 三次样条插值； 4. INTER\_AREA: 区域插值； 5. INTER\_LANCZOS4: 8x8像素邻域内的Lanczos插值； 6. INTER\_LINEAR\_EXACT: 精确bit双线性插值； 7. INTER\_NEAREST\_EXACT: 精确bit最近邻插值； 8. INTER\_MAX: 掩码插值； 9. WARP\_FILL\_OUTLIERS: 填充满放大后的目标图像，如果目标图像中对应原图的像素已经溢出，则将其置0； 10. WARP\_INVERSE\_MAP: 反变换； |
| nDestImage | REFERENCE TO ULINT | 输出指定大小的图像，类型与输入相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Resize



目标尺寸缩放，通过定义目标图片的宽与高，调整输入图片的大小。若是调整后的图片面积小于目标图片的大小，则边框区域像素值为0来补全图片。

**Syntax**

Definition:

FUNCTION F\_CV\_Resize: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nWidth : DINT := 1133;

nHeight : DINT := 426;

eInterpolation : E\_CvInterpolationFlags := E\_CvInterpolationFlags.INTER\_LINEAR;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

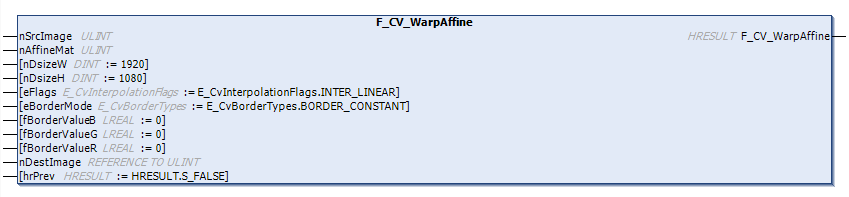
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片 |
| nWidth | DINT | 宽度，输出图像大小 |
| nHeight | DINT | 高度，输出图像大小 |
| eInterpolation | E\_CvInterpolationFlags | 处理像素的插值方法   1. INTER\_NEAREST: 最近邻插值； 2. INTER\_LINEAR: 线性插值； 3. INTER\_CUBIC: 三次样条插值； 4. INTER\_AREA: 区域插值； 5. INTER\_LANCZOS4: 8x8像素邻域内的Lanczos插值； 6. INTER\_LINEAR\_EXACT: 精确bit双线性插值； 7. INTER\_NEAREST\_EXACT: 精确bit最近邻插值； 8. INTER\_MAX: 掩码插值； 9. WARP\_FILL\_OUTLIERS: 填充满放大后的目标图像，如果目标图像中对应原图的像素已经溢出，则将其置0； 10. WARP\_INVERSE\_MAP: 反变换； |
| nDestImage | REFERENCE TO ULINT | 输出指定大小的图像，类型与输入相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_WarpAffine



图像仿射变换，输入一张图像与一个仿射变换的矩阵，将图像进行仿射变换，从而得到变换后的图像。通过定义输出图像的宽和高，从而改变原图像的长宽比。边框区域将按照定义的颜色填补。

**Syntax**

Definition:

FUNCTION F\_CV\_WarpAffine: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nAffineMat : ULINT;

nDsizeW : DINT := 1920;

nDsizeH : DINT := 1080;

eFlags : E\_CvInterpolationFlags := E\_CvInterpolationFlags.INTER\_LINEAR;

eBorderMode : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_CONSTANT;

fBorderValueB : LREAL := 0;

fBorderValueG : LREAL := 0;

fBorderValueR : LREAL := 0;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

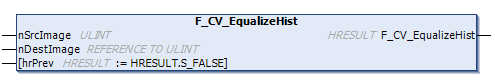
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片 |
| nAffineMat | ULINT | 变换矩阵 |
| nDsizeW | DINT | 输出图像的宽 |
| nDsizeH | DINT | 输出图像的高 |
| eFlags | E\_CvInterpolationFlags | 插值方法   1. INTER\_NEAREST: 最近邻插值； 2. INTER\_LINEAR: 线性插值； 3. INTER\_CUBIC: 三次样条插值； 4. INTER\_AREA: 区域插值； 5. INTER\_LANCZOS4: 8x8像素邻域内Lanczos插值； 6. INTER\_LINEAR\_EXACT: 精确bit双线性插值； 7. INTER\_NEAREST\_EXACT: 精确bit最近邻插值； 8. INTER\_MAX: 掩码插值； 9. WARP\_FILL\_OUTLIERS: 填充满放大后的目标图像，如果目标图像中对应原图的像素已经溢出，则将其置0； 10. WARP\_INVERSE\_MAP: 反变换； |
| eBorderMode | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_WRAP: 如 cdefgh|abcdefgh|abcdefg 用另一边的像素来补偿填充； 2. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 3. BORDER\_REPLICATE: 如aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 4. BORDER\_REFLECT: 如fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 5. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 6. BORDER\_DEFAULT: 如gfedcb|abcdefgh|gfedcba以边界为对称轴反射复制像素； 7. BORDER\_ISOLATED: 不看ROI之外部分； |
| fBorderValueB | LREAL | 若边框类型为BORDER\_CONSTANT，边界值的蓝色 |
| fBorderValueG | LREAL | 若边框类型为BORDER\_CONSTANT，边界值的绿色 |
| fBorderValueR | LREAL | 若边框类型为BORDER\_CONSTANT，边界值的红色 |
| nDestImage | REFERENCE TO ULINT | 输出指定大小的图像，类型与输入相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_EqualizeHist



直方图均衡化，输入一张单通道图片，通过计算图像的直方图，对其进行均衡化处理，步骤如下，

1. 对输入图像计算直方图；
2. 对直方图进行归一化处理，使直方图区间之和为255；
3. 计算直方图的微积分；
4. 使用微积分的结果作为查找表格，处理输入的图片；
5. 对亮度进行归一化处理，并提高图像的对比度；

**Syntax**

Definition:

FUNCTION F\_CV\_EqualizeHist: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

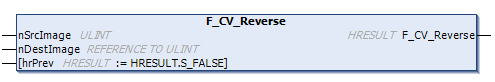
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 8位单通道图像 |
| nDestImage | REFERENCE TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Reverse



图像反色，输入一张图片，对图像求反色，如将白色像素变成黑色像素、将黑色像素变成白色像素。

**Syntax**

Definition:

FUNCTION F\_CV\_Reverse: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

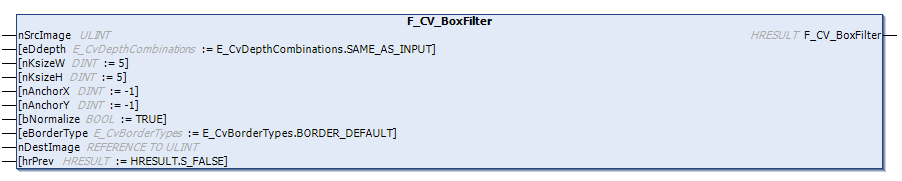
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 源图像为单通道或三通道图像。 |
| nDestImage | REFERENCE TO ULINT | 处理后的图片，与输入大小和类型相同。 |
| hrPrev | HRESULT | 算子的初始执行状况。 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BoxFilter



方形滤波，使用方框过滤器模糊图像。当使用归一化的方形滤波，则滤波矩阵除以模糊内核的面积；当使用非归一化的方形滤波，则不对滤波矩阵处理，可用于计算每个像素邻域上的各种特性，如图像导数的协方差矩阵（用于密集光流算法等）。

**Syntax**

Definition:

FUNCTION F\_CV\_BoxFilter: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

eDdepth : E\_CvDepthCombinations := E\_CvDepthCombinations.SAME\_AS\_INPUT;

nKsizeW : DINT := 5;

nKsizeH : DINT := 5;

nAnchorX : DINT := -1;

nAnchorY : DINT := -1;

bNormalize : BOOL := TRUE;

eBorderType : E\_CvBorderTypes := E\_CvBorderTypes.BORDER\_DEFAULT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

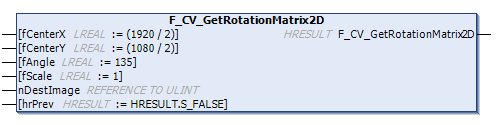
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像 |
| eDdepth | E\_CvDepthCombinations | 输出图像深度   1. 输入深度：8U，输出深度：-1/16S/32F/64F； 2. 输入深度：16U/16S，输出深度：-1/32F/64F； 3. 输入深度：32F，输出深度：-1/32F/64F； 4. 输入深度：64F，输出深度：-1/64F；   注意：当eDdepth=-1时，输出图像将具有与源相同的深度。 |
| nKsizeW | DINT | 模糊内核宽，必须是正数和奇数 |
| nKsizeH | DINT | 模糊内核高，必须是正数和奇数 |
| nAnchorX | DINT | 锚点x坐标；默认值-1，表示锚点位于内核中心 |
| nAnchorY | DINT | 锚点y坐标；默认值-1，表示锚点位于内核中心 |
| bNormalize | BOOL | 指定内核是否按其面积进行规范化 |
| eBorderType | E\_CvBorderTypes | 边框模式用于图像外部的像素   1. BORDER\_CONSTANT: 如 iiiiii|abcdefgh|iiiiiii，需要指定值i，即就是边沿像素用i替换； 2. BORDER\_REPLICATE: 如 aaaaaa|abcdefgh|hhhhhhh，复制边界像素； 3. BORDER\_REFLECT: 如 fedcba|abcdefgh|hgfedcb，反射复制边界像素，镜像复制； 4. BORDER\_TRANSPARENT: 如 uvwxyz|abcdefgh|ijklmno，推导赋值； 5. BORDER\_DEFAULT: 如 gfedcb|abcdefgh|gfedcba，以边界为对称轴反射复制像素； 6. BORDER\_ISOLATED: 不看ROI之外部分； |
| DestImage | REFERENCE TO ULINT | 处理后的图片，与输入大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_GetRotationMatrix2D



旋转矩阵2D，根据图像的旋转角度、旋转中心、缩放因子，计算仿射变换的矩阵。配合图像仿射变换的函数、用于图片的仿射变换。

**Syntax**

Definition:

FUNCTION F\_CV\_GetRotationMatrix2D: HRESULT

VAR\_INPUT

fCenterX : LREAL := 1920/2;

fCenterY : LREAL := 1080/2;

fAngle : LREAL := 135;

fScale : LREAL := 1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| fCenterX | LREAL | 源图像中的旋转中心x坐标 |
| fCenterY | LREAL | 源图像中的旋转中心y坐标 |
| fAngle | LREAL | 旋转角度（以度为单位），正值表示逆时针旋转（坐标原点假定为左上角） |
| fScale | LREAL | 各向同性比例因子 |
| DestImage | REFERENCE  TO ULINT | 输出矩阵 (图像格式) |
| hrPrev | HRESULT | 算子的初始执行状况 |

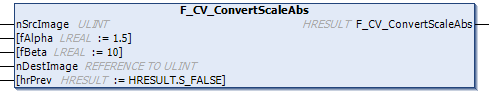
**Return value**

HRESULT

**Further information**

### 图像运算

#### F\_CV\_ConvertScaleAbs



灰度扩展，在输入图像的每个像素上，按顺序执行三个操作：缩放、取绝对值、转换为无符号8位类型，在多通道数组的情况下，独立处理每个通道, 若像素值-255 <= nSrcImage\*fAlpha+fBeta <= 255，取绝对值；若像素值 nSrcImage\*fAlpha+fBeta > 255 或者 nSrcImage\*fAlpha+fBeta < -255，取255，不可溢出。

**Syntax**

Definition:

FUNCTION F\_CV\_ConvertScaleAbs: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fAlpha : LREAL := 1.5;

fBeta : LREAL := 10;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

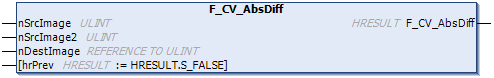
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像 |
| fAlpha | LREAL | 比例因子 |
| fBeta | LREAL | 偏移量 |
| nDestImage | REFERENCE TO ULINT | 输出图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_AbsDiff



绝对差，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的绝对差，在多通道图像的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_AbsDiff: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

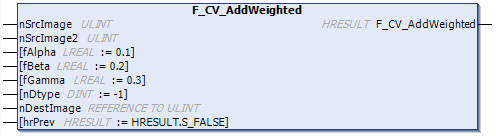
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDestImage | REFERENCE TO ULINT | 输出图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_AddWeighted



加权求和，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的加权和，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_AddWeighted: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

fAlpha : LREAL := 0.1;

fBeta : LREAL := 0.2;

fGamma : LREAL := 0.3;

nDtype : DINT := -1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

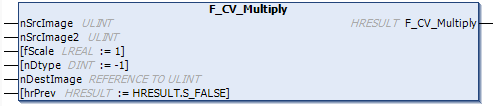
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| fAlpha | LREAL | 第一个图像的权重 |
| fBeta | LREAL | 第二个图像的权重 |
| fGamma | LREAL | 标量加到每个和上 |
| nDtype | DINT | 输出图像的可选深度，-1与输入图片相同 |
| nDestImage | REFERENCE TO ULINT | 输出图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Multiply



相乘，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的按比例缩放后的乘积，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_Multiply: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

fScale : LREAL := 1;

nDtype : DINT := -1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

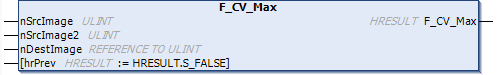
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| fScale | LREAL | 比例因子 |
| nDtype | DINT | 输出图像的可选深度 |
| nDestImage | REFERENCE TO ULINT | 输出图像，大小类型与图一相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Max



最大值，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的最大值，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_Max: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

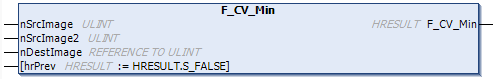
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDestImage | REFERENCE TO ULINT | 输出图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Min



最小值，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的最小值，在多通道图片的情况下，每个通道都是独立处理的

**Syntax**

Definition:

FUNCTION F\_CV\_Min: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

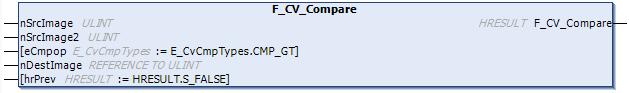
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDestImage | REFERENCE TO ULINT | 输出图像 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Compare



比较，输入两张相同大小、相同类型的图片，按照选定的比较规则，逐个比较两个图之间、相同位置的像素值，在多通道图片的情况下，每个通道都是独立处理的

**Syntax**

Definition:

FUNCTION F\_CV\_Compare: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

eCmpop : E\_CvCmpTypes := E\_CvCmpTypes.CMP\_GT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

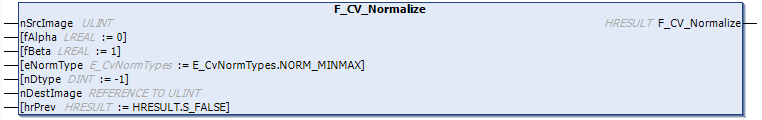
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小与第一张相同 |
| eCmpop | E\_CvCmpTypes | 运算模式   1. CMP\_EQ: 相等； 2. CMP\_GT: 大于； 3. CMP\_GE: 大于等于； 4. CMP\_LT: 小于； 5. CMP\_LE: 小于等于； 6. CMP\_NE: 不相等； |
| nDestImage | REFERENCE  TO ULINT | 输出图像 8U类型，与输入相同的大小和相同数量的通道 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Normalize



范数，输入图像为单通道，根据归一化的选择类型，对图片进行范数化操作。该函数对输入图像元素进行缩放和移位归一化。当规范化类型为NORM\_INF、NORM\_L1或NORM\_L2时，函数将使用fAlpha进行处理；当归一化类型为NORM\_MINMAX时，函数将使用fAlpha作为最小值、fBeta作为最大值进行处理。

**Syntax**

Definition:

FUNCTION F\_CV\_Normalize: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fAlpha : LREAL := 0;

fBeta : LREAL := 1;

eNormType : E\_CvNormTypes := E\_CvNormTypes.NORM\_MINMAX;

nDtype : DINT := -1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

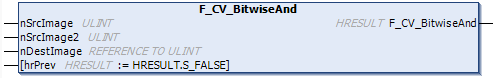
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的单通道图像 |
| fAlpha | LREAL | 如果eNormType为NORM\_MINMAX ，则alpha为最小值或最大值；如果eNormType为其他类型，则为归一化要乘的系数。 |
| fBeta | LREAL | 如果eNormType为NORM\_MINMAX，则beta为最小值或最大值；如果eNormType为其他类型，beta被忽略，此处不会被用到，一般传入0。 |
| eNormType | E\_CvNormTypes | 归一化类型   1. NORM\_L1: L1-范数，绝对值和； 2. NORM\_L2: L2-范数，平方和的开方； 3. NORM\_INF: 无穷范数，绝对值的最大值； 4. NORM\_MINMAX: 分别为归一化后的最小值、最大值， 函数会自动判断哪个为最大值，最小值； |
| nDtype | DINT | 如果取负值时，输入与输出同样的类型；  否则，输入与输出有同样的通道数 |
| nDestImage | REFERENCE  TO ULINT | 输出图像，与输入尺寸相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BitwiseAnd



位与，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的位与，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_BitwiseAnd: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

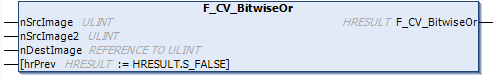
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDestImage | REFERENCE TO ULINT | 输出图像，与输入尺寸相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BitwiseOr



位或，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的位或，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_BitwiseOr: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDestImage | REFERENCE TO ULINT | 输出图像，与输入尺寸相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BitwiseNot



位非，输入一张图片，计算每个像素的反转，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_BitwiseNot: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

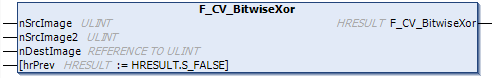
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nDestImage | REFERENCE TO ULINT | 输出图像，与输入尺寸相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_BitwiseXor



位异或，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置像素的位异或，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_BitwiseXor: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

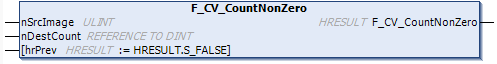
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDestImage | REFERENCE TO ULINT | 输出图像，与输入尺寸相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_CountNonZero



非零元素数量，输入一张单通道图像，统计所有非零像素的个数，返回一个统计值。

**Syntax**

Definition:

FUNCTION F\_CV\_CountNonZero: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nDestCount : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

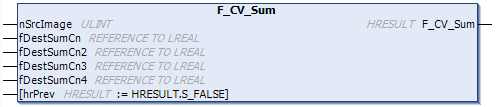
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入的单通道图像 |
| nDestCount | REFERENCE TO DINT | 非零元素的数量 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Sum



元素和，输入一张图片，最多4个通道，计算每个通道中像素值的总和。

**Syntax**

Definition:

FUNCTION F\_CV\_Sum: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fDestSumCn : REFERENCE TO LREAL;

fDestSumCn2 : REFERENCE TO LREAL;

fDestSumCn3 : REFERENCE TO LREAL;

fDestSumCn4 : REFERENCE TO LREAL;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

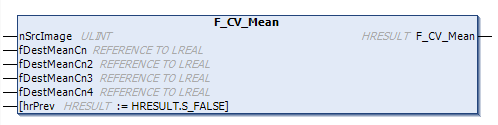
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像，必须具有1到4个通道 |
| fDestSumCn | REFERENCE TO LREAL | 图像第一个通道的像素值总和 |
| fDestSumCn2 | REFERENCE TO LREAL | 图像第二个通道的像素值总和 |
| fDestSumCn3 | REFERENCE TO LREAL | 图像第三个通道的像素值总和 |
| fDestSumCn4 | REFERENCE TO LREAL | 图像第四个通道的像素值总和 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Mean



计算图像均值，计算图像元素的平均值，独立计算每个通道的元素的平均值并返回。

**Syntax**

Definition:

FUNCTION F\_CV\_Mean: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fDestMeanCn : REFERENCE TO LREAL;

fDestMeanCn2 : REFERENCE TO LREAL;

fDestMeanCn3 : REFERENCE TO LREAL;

fDestMeanCn4 : REFERENCE TO LREAL;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

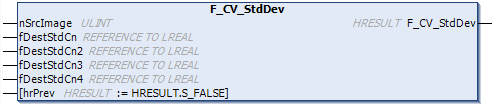
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像，必须具有1到4个通道 |
| fDestMeanCn | REFERENCE TO LREAL | 图像第一个通道的元素平均值 |
| fDestMeanCn2 | REFERENCE TO LREAL | 图像第二个通道的元素平均值 |
| fDestMeanCn3 | REFERENCE TO LREAL | 图像第三个通道的元素平均值 |
| fDestMeanCn4 | REFERENCE TO LREAL | 图像第四个通道的元素平均值 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_StdDev



方差，计算图像像素的标准差，独立计算每个通道的像素标准差并返回。

**Syntax**

Definition:

FUNCTION F\_CV\_StdDev: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fDestStdCn : REFERENCE TO LREAL;

fDestStdCn2 : REFERENCE TO LREAL;

fDestStdCn3 : REFERENCE TO LREAL;

fDestStdCn4 : REFERENCE TO LREAL;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

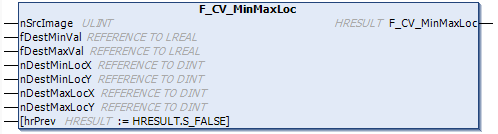
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图像，必须具有1到4个通道 |
| fDestStdCn | REFERENCE TO LREAL | 图像第一个通道的像素值标准差 |
| fDestStdCn2 | REFERENCE TO LREAL | 图像第二个通道的像素值标准差 |
| fDestStdCn3 | REFERENCE TO LREAL | 图像第三个通道的像素值标准差 |
| fDestStdCn4 | REFERENCE TO LREAL | 图像第四个通道的像素值标准差 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_MinMaxLoc



最值定位，输入一张单通道的图片，获取图像内最小和最大的像素值，以及他们在图像中的位置，坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向。

**Syntax**

Definition:

FUNCTION F\_CV\_MinMaxLoc: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

fDestMinVal : REFERENCE TO LREAL;

fDestMaxVal : REFERENCE TO LREAL;

nDestMinLocX : REFERENCE TO DINT;

nDestMinLocY : REFERENCE TO DINT;

nDestMaxLocX : REFERENCE TO DINT;

nDestMaxLocY : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

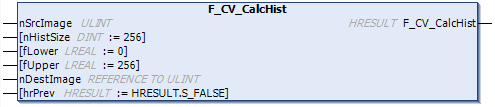
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入图片，单通道 |
| fMinVal | REFERENCE TO LREAL | 最小的像素值 |
| fMaxVal | REFERENCE TO LREAL | 最大的像素值 |
| nMinLocX | REFERENCE TO DINT | 最小的像素值在图像中x坐标 |
| nMinLocY | REFERENCE TO DINT | 最小的像素值在图像中y坐标 |
| nMaxLocX | REFERENCE TO DINT | 最大的像素值在图像中x坐标 |
| nMaxLocY | REFERENCE TO DINT | 最大的像素值在图像中y坐标 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_CalcHist



直方图，输入一张单通道的图片，根据图像的像素值，计算绘制直方图所需的数据，通过调用图像绘制中的绘制直方图功能，从而展示计算的数据。

**Syntax**

Definition:

FUNCTION F\_CV\_CalcHist: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nHistSize : DINT := 256;

fLower : LREAL := 0;

fUpper : LREAL := 256;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

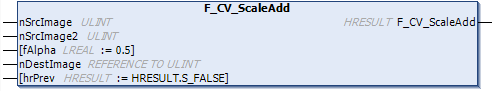
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入单通道的图片 |
| nHistSize | DINT | 每个维度中的直方图大小 |
| fLower | LREAL | 每个维度中，bin的下边界 |
| fUpper | LREAL | 每个维度中，bin的上边界 |
| nDestImage | REFERENCE TO ULINT | 返回绘制直方图所需的数据 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_ScaleAdd



加权后相加，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置的像素缩放后的和，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_ScaleAdd: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

fAlpha : LREAL := 0.5;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

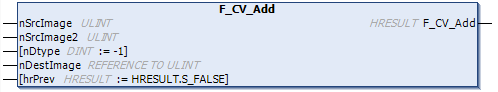
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| fAlpha | LREAL | 第一张图片的比例因子 |
| nDestImage | REFERENCE TO ULINT | 输出图像，与输入第一张图片大小和类型相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Add



加，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置的像素和，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_Add: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDtype : DINT := -1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

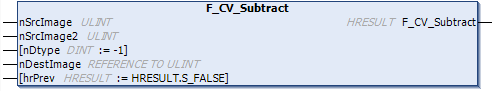
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDtype | DINT | 输出图像的深度 |
| nDestImage | REFERENCE TO ULINT | 输出图像，大小和通道数与输入相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_Subtract



减，输入两张相同大小、相同类型的图片，计算两个图之间、相同位置的像素差，在多通道图片的情况下，每个通道都是独立处理的。

**Syntax**

Definition:

FUNCTION F\_CV\_Subtract: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImage2 : ULINT;

nDtype : DINT := -1;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入第一张图片 |
| nSrcImage2 | ULINT | 输入第二张图片，大小类型与输入的第一张图片相同 |
| nDtype | DINT | 输出图像的深度 |
| nDestImage | REFERENCE TO ULINT | 输出图像，大小和通道数与输入相同 |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_MatchTemplate



模板匹配，输入一张单通道的模板图和一张单通道的测试图，在测试图中根据模板图的大小创建滑动窗口来滑动，使用指定的匹配方法，将模板图像与测试图像中滑动窗口重叠的区域进行比较。比较后，返回的结果图需要使用最值定位的函数找到最佳的匹配区域，当使用TM\_SQDIFF方法，将使用最小值选取最佳匹配；当使用TM\_CCORR或TM\_CCOEFF方法，将使用最大值来选取最佳匹配。

**Syntax**

Definition:

FUNCTION F\_CV\_MatchTemplate: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImageTempl : ULINT;

eMethod : E\_CvTemplateMatchModes :=

E\_CvTemplateMatchModes.TM\_CCOEFF\_NORMED;

nDestImage : REFERENCE TO ULINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

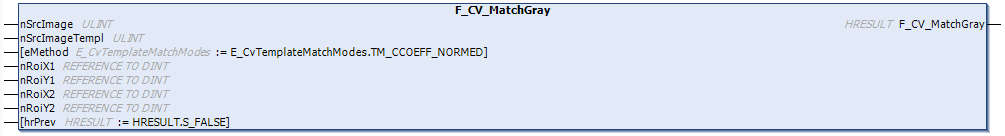
|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入测试图片，单通道8位或32位浮点 |
| nSrcImageTempl | ULINT | 输入模板图片，单通道，它必须不大于测试图片大小，并且具有相同的数据类型 |
| eMethod | E\_CvTemplateMatchModes | 指定的匹配方法   1. TM\_SQDIFF: 平方差匹配法，使用平方差来进行匹配，最好匹配为0。而若匹配越差，匹配值则越大。(说明最黑的地方是匹配最好的地方)； 2. TM\_SQDIFF\_NORMED: 归一化平方差匹配法，最好匹配为0； 3. TM\_CCORR: 相关匹配法，这类方法用模板和图像间的乘法操作，所以较大的数表示匹配程序较高，0标识最坏的匹配效果； 4. TM\_CCORR\_NORMED: 归一化相关匹配法，最坏匹配为0； 5. TM\_CCOEFF: 系数匹配法，最好匹配为1； 6. TM\_CCOEFF\_NORMED: 归一化相关系数匹配法，最好匹配为1； |
| nDestImage | REFERENCE TO ULINT | 输出比较结果的映射图像，单通道8位或32位浮点。如果图像尺寸是WxH而temp1尺寸是wxh，则此参数result一定是 (W-w+1)x(H-h+1) |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

#### F\_CV\_MatchGray



模板识别，输入一张单通道的模板图和一张单通道的测试图，在测试图中根据模板图的大小创建滑动窗口来滑动，使用指定的匹配方法，将模板图像与测试图像中滑动窗口重叠的区域进行比较，返回最佳的匹配区域的矩形框坐标，坐标原点为图像左上角，向右为x轴的正方向，向下为y轴的正方向，当使用TM\_SQDIFF方法，将使用最小值选取最佳匹配；当使用TM\_CCORR或TM\_CCOEFF方法，将使用最大值来选取最佳匹配。

**Syntax**

Definition:

FUNCTION F\_CV\_MatchGray: HRESULT

VAR\_INPUT

nSrcImage : ULINT;

nSrcImageTempl : ULINT;

eMethod : E\_CvTemplateMatchModes :=

E\_CvTemplateMatchModes.TM\_CCOEFF\_NORMED;

nRoiX1 : REFERENCE TO DINT;

nRoiY1 : REFERENCE TO DINT;

nRoiX2 : REFERENCE TO DINT;

nRoiY2 : REFERENCE TO DINT;

hrPrev : HRESULT := HRESULT.S\_FALSE;

END\_VAR

**Inputs**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| nSrcImage | ULINT | 输入测试图片，单通道8位或32位浮点 |
| nSrcImageTempl | ULINT | 输入模板图片，单通道，它必须不大于测试图片大小，并且具有相同的数据类型 |
| eMethod | E\_CvTemplateMatchModes | 指定的匹配方法   1. TM\_SQDIFF: 平方差匹配法，使用平方差来进行匹配，最好匹配为0。而若匹配越差，匹配值则越大。(说明最黑的地方是匹配最好的地方)； 2. TM\_SQDIFF\_NORMED: 归一化平方差匹配法，最好匹配为0； 3. TM\_CCORR: 相关匹配法，这类方法用模板和图像间的乘法操作，所以较大的数表示匹配程序较高，0标识最坏的匹配效果； 4. TM\_CCORR\_NORMED: 归一化相关匹配法，最坏匹配为0； 5. TM\_CCOEFF: 系数匹配法，最好匹配为1； 6. TM\_CCOEFF\_NORMED: 归一化相关系数匹配法，最好匹配为1； |
| nRoiX1 | REFERENCE TO DINT | 最佳匹配区域的矩形框左上角坐标x |
| nRoiY1 | REFERENCE TO DINT | 最佳匹配区域的矩形框左上角坐标y |
| nRoiX2 | REFERENCE TO DINT | 最佳匹配区域的矩形框右下角坐标x |
| nRoiY2 | REFERENCE TO DINT | 最佳匹配区域的矩形框右下角坐标y |
| hrPrev | HRESULT | 算子的初始执行状况 |

**Return value**

HRESULT

**Further information**

## 时序算子库（昊庭）