目录

[1. 概述](#_Toc16561_WPSOffice_Level1) [1](#_Toc16561_WPSOffice_Level1)

[1.1. 概述](#_Toc5912_WPSOffice_Level2) [1](#_Toc5912_WPSOffice_Level2)

[1.2. 参考域说明](#_Toc4783_WPSOffice_Level2) [2](#_Toc4783_WPSOffice_Level2)

[1.2.1. 参考域说明](#_Toc5912_WPSOffice_Level3) [2](#_Toc5912_WPSOffice_Level3)

[1.2.2. 数据类型参考域](#_Toc4783_WPSOffice_Level3) [2](#_Toc4783_WPSOffice_Level3)

[2. API参考](#_Toc5912_WPSOffice_Level1) [2](#_Toc5912_WPSOffice_Level1)

[2.1. HYKR\_Init\_GPU](#_Toc21394_WPSOffice_Level2) [2](#_Toc21394_WPSOffice_Level2)

[2.2. HYKR\_SetParam\_GPU](#_Toc3618_WPSOffice_Level2) [3](#_Toc3618_WPSOffice_Level2)

[2.3. HYKR\_KnobRecog\_GPU](#_Toc24124_WPSOffice_Level2) [4](#_Toc24124_WPSOffice_Level2)

[2.4. HYKR\_Uninit\_GPU](#_Toc18494_WPSOffice_Level2) [4](#_Toc18494_WPSOffice_Level2)

[2.5. HYL\_MatchInit](#_Toc11832_WPSOffice_Level2) [5](#_Toc11832_WPSOffice_Level2)

[2.6. HYL\_TrainTemplateFromMask](#_Toc14202_WPSOffice_Level2) [5](#_Toc14202_WPSOffice_Level2)

[2.7. HYL\_SaveTDescriptorsGroup](#_Toc13460_WPSOffice_Level2) [6](#_Toc13460_WPSOffice_Level2)

[2.8. HYL\_GetTemplateFromText](#_Toc21332_WPSOffice_Level2) [6](#_Toc21332_WPSOffice_Level2)

[2.9. HYL\_GetDashboard](#_Toc21209_WPSOffice_Level2) [7](#_Toc21209_WPSOffice_Level2)

[2.10. HYL\_MatchUninit](#_Toc10031_WPSOffice_Level2) [8](#_Toc10031_WPSOffice_Level2)

[3. 结构体说明](#_Toc4783_WPSOffice_Level1) [8](#_Toc4783_WPSOffice_Level1)

[3.1. 图像格式](#_Toc16111_WPSOffice_Level2) [8](#_Toc16111_WPSOffice_Level2)

[3.1.1. 算法库图像格式](#_Toc21394_WPSOffice_Level3) [8](#_Toc21394_WPSOffice_Level3)

[3.1.2. 刚体匹配图像格式](#_Toc3618_WPSOffice_Level3) [9](#_Toc3618_WPSOffice_Level3)

[3.2. 输出结果的结构体](#_Toc20760_WPSOffice_Level2) [10](#_Toc20760_WPSOffice_Level2)

[3.2.1. 单个结果的结构体](#_Toc24124_WPSOffice_Level3) [10](#_Toc24124_WPSOffice_Level3)

[3.2.2. 区域的结构体](#_Toc18494_WPSOffice_Level3) [11](#_Toc18494_WPSOffice_Level3)

[3.2.3. 坐标的结构体](#_Toc11832_WPSOffice_Level3) [11](#_Toc11832_WPSOffice_Level3)

# 概述

## 概述

把手算法是四川华雁信息产业股份有限公司提供的用于变电站把手识别的算法模块。

## 参考域说明

### 参考域说明

本说明文档选用了8个描述API的相关信息，它们的作用如下表1-1所示。

表1-1 API参考域说明

|  |  |
| --- | --- |
| 参考域 | 含义 |
| 1 目的 | 描述相应 API 的主要功能 |
| 2 语法 | 列出调用 API 需要包括的头文件以及 API 的原形声明 |
| 3 参数 | 列出了 API 中调用的参数，参数类型以及参数属性说明 |
| 4 描述 | 简要的描述 API 的工作过程 |
| 5 返回值 | 描述了 API 返回的值及其含义 |
| 6 需求 | 描述 API 包含的头文件以及其需要连接的库 |
| 7 注意 | 使用 API 时需要注意的点 |
| 8 举例 | 描述如何使用 API |

### 数据类型参考域

本说明文档采用了 5 个参考域描述数据类型的相关信息，它们的作用如下表 1-2 所示。

表1-2 参考域描述数据类型

|  |  |
| --- | --- |
| 参考域 | 含义 |
| 1 说明 | 描述该数据类型的主要功能 |
| 2 含义 | 列出数据类型的定义 |
| 3 成员 | 列出该数据类型包含的成员 |
| 4 注意事项 | 列出了使用该数据类型的注意点 |
| 5 相关数据类型及接口 | 列出了与该数据类型相关的数据类型以及接口 |

# API参考

## HYKR\_Init\_GPU

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 初始化句柄 | | |
| 2 语法 | 需包含头文件"KnobRecog\_GPU.h"  调用语法 HYKR\_Init\_GPU(void \*hMemMgr, void \*\*pMRHandle) | | |
| 3参数 | void \*hMemMgr | | 设为NULL |
| void \*\*pMRHandle | | 模型空间句柄 |
| 4 描述 | 初始化句柄pMRHandle | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 调用需要包含头文件"KnobRecog\_GPU.h"  需要包含库文件“yolo\_cpp\_dll\_gpu.dll”“yolo\_cpp\_dll\_gpu.lib”  “KnobRecog\_GPU.dll”“KnobRecog\_GPU.lib” | | |
| 7 注意 |  | | |
| 8 举例 | void \*hTLHandle=NULL;  HYKR\_Init\_GPU(NULL,&hTLHandle); | | |

## HYKR\_SetParam\_GPU

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 加载模型到内存中 | | |
| 2 语法 | 需包含头文件"KnobRecog\_GPU.h"  调用语法 HYKR\_SetParam\_GPU(void \*hMRHandle,char \*cfgfile, char \*weightfile,float Thresh,int gpu\_index,int w,int h); | | |
| 3参数 | void \*hMRHandle | | 模型空间句柄 |
| char \*cfgfile | | 模型网络结构文件 |
| char \*weightfile | | 模型网络权重文件 |
| float Thresh | | 模型阈值 |
| int gpu\_index | | 调用的GPU显卡序号 |
| int w | | 输入图片宽 |
| int h | | 输入图片高 |
| 4 描述 | 加载模型到内存中 | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 需包含头文件"KnobRecog\_GPU.h"  需要包含库文件“yolo\_cpp\_dll\_gpu.dll”“yolo\_cpp\_dll\_gpu.lib”  “KnobRecog\_GPU.dll”“KnobRecog\_GPU.lib” | | |
| 7 注意 | w和h要设置为图片尺寸，不可以为0 | | |
| 8 举例 | void \*hTLHandle=NULL;  char \*cfgfile="\*.cfg";  char \*weightfile="\*.weights";// 模型  float thresh=0.1;  int gpu\_index = 0;  int w=输入图片宽;  int h=输入图片高;  HYKR\_Init\_GPU(NULL,&hTLHandle);  HYKR\_SetParam\_GPU(hTLHandle,cfgfile,weightfile,thresh,gpu\_index,w,h); | | |

## HYKR\_KnobRecog\_GPU

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 检测输入图像中的目标 | | |
| 2 语法 | 需包含头文件"KnobRecog\_GPU.h"  调用语法HYKR\_KnobRecog\_GPU(void \*hMRHandle,KR\_IMAGES \*pImg , HYKR\_RESULT\_LIST \*pResultList) | | |
| 3参数 | void \*hMRHandle | | 模型空间句柄 |
| KR\_IMAGES \*pImg | | 输入的待测图片 |
| HYKR\_RESULT\_LIST \*pResultList | | 结果输出 |
| 4 描述 | 检测输入图像中的目标 | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 需包含头文件"KnobRecog\_GPU.h"  需要包含库文件“yolo\_cpp\_dll\_gpu.dll”“yolo\_cpp\_dll\_gpu.lib”  “KnobRecog\_GPU.dll”“KnobRecog\_GPU.lib” | | |
| 7 注意 |  | | |
| 8 举例 | HYKR\_KnobRecog\_GPU(hTLHandle,&imgs,&resultlist) | | |

## HYKR\_Uninit\_GPU

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 释放内存 | | |
| 2 语法 | 需包含头文件"KnobRecog\_GPU.h"  调用语法HYKR\_Uninit\_GPU(void \*hMRHandle) | | |
| 3参数 | void \*hMRHandle | | 模型空间句柄 |
| 4 描述 | 释放内存 | | |
| 5 返回值 |  |  | |
|  |  | |
| 6 需求 | 需包含头文件"KnobRecog\_GPU.h"  需要包含库文件“yolo\_cpp\_dll\_gpu.dll”“yolo\_cpp\_dll\_gpu.lib”  “KnobRecog\_GPU.dll”“KnobRecog\_GPU.lib” | | |
| 7 注意 |  | | |
| 8 举例 | HYKR\_Uninit\_GPU(hTLHandle) | | |

## HYL\_MatchInit

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 刚体匹配初始化句柄 | | |
| 2 语法 | 需包含头文件"HYL\_MatchRigidBody.h"  调用语法 HYL\_MatchInit(MHandle hMemMgr, MHandle \*pHYEDHandle) | | |
| 3参数 | void \*hMemMgr | | 设为NULL |
| void \*\*pMRHandle | | 输出的内部操作句柄 |
| 4 描述 | 初始化句柄pMRHandle | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 调用需要包含头文件"HYL\_MatchRigidBody.h"  需要包含库文件“MatchRigidBody.dll”“MatchRigidBody.lib” | | |
| 7 注意 |  | | |
| 8 举例 | void \*MHandle=NULL;  HYL\_MatchInit(NULL, &MHandle); | | |

## HYL\_TrainTemplateFromMask

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 获取目标对象描述符 | | |
| 2 语法 | 需包含头文件"HYL\_MatchRigidBody.h"  调用语法HYL\_TrainTemplateFromMask(MHandle hHYMRDHandle,HYL\_PIMAGES pImage,HYL\_PIMAGES pMask,MChar \*pClassName,MLong lParma) | | |
| 3参数 | MHandle hHYMRDHandle | | 库函数操作句柄 |
| HYL\_PIMAGES pImage | | 目标图像 |
| HYL\_PIMAGES pMask | | Mask模板图像 |
| MChar \*pClassName | | 模板类名 |
| MLong lParma | | 训练的方式。0不对图像进行旋转处理，1对图像进行旋转处理 |
| 4 描述 | 根据目标训练图像和mask模板图像进行刚体匹配训练，获取对象描述符 | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 调用需要包含头文件"HYL\_MatchRigidBody.h"  需要包含库文件“MatchRigidBody.dll”“MatchRigidBody.lib” | | |
| 7 注意 |  | | |
| 8 举例 | // pOrgImg和maskImage为调用opencv获取的图像  HYL\_IMAGES wy\_testImage = {0}, wy\_maskImage = {0};  wy\_testImage.lWidth = pOrgImg->width;  wy\_testImage.lHeight = pOrgImg->height;  wy\_testImage.pixelArray.chunky.lLineBytes = pOrgImg->widthStep;  wy\_testImage.pixelArray.chunky.pPixel = pOrgImg->imageData;  wy\_testImage.lPixelArrayFormat = HYL\_IMAGE\_BGR;  //mask  wy\_maskImage.lWidth = maskImage->width;  wy\_maskImage.lHeight = maskImage->height;  wy\_maskImage.pixelArray.chunky.lLineBytes = maskImage->widthStep;  wy\_maskImage.pixelArray.chunky.pPixel = maskImage->imageData;  wy\_maskImage.lPixelArrayFormat = HYL\_IMAGE\_GRAY;  if (0 != HYL\_TrainTemplateFromMask (MHandle, &wy\_testImage, &wy\_maskImage, "OK", 0))//训练图片，得到模板  {  printf("HYAMR\_TrainTemplateFromMask error!\n");  goto EXT;  } | | |

## HYL\_SaveTDescriptorsGroup

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 存储刚体匹配训练获得的描述符信息 | | |
| 2 语法 | 需包含头文件"HYL\_MatchRigidBody.h"  调用语法HYL\_SaveTDescriptorsGroup (MHandle hHYMRDHandle,const char \*path) | | |
| 3参数 | MHandle hHYMRDHandle | | 库函数操作句柄 |
| const char \*path | | 存储文件路径 |
| 4 描述 | 保存描述符信息 | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 调用需要包含头文件"HYL\_MatchRigidBody.h"  需要包含库文件“MatchRigidBody.dll”“MatchRigidBody.lib” | | |
| 7 注意 |  | | |
| 8 举例 | #define MR\_READ\_FILE\_PARA "../MatchRigidBody.dat"  if (0 != HYL\_SaveTDescriptorsGroup(MHandle, MR\_READ\_FILE\_PARA))  {  printf("HYAMR\_SaveTDescriptorsGroup error!\n");  } | | |

## HYL\_GetTemplateFromText

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 读取刚体匹配训练获得的描述符信息 | | |
| 2 语法 | 需包含头文件"HYL\_MatchRigidBody.h"  调用语法HYL\_GetTemplateFromText (MHandle hHYMRDHandle,char\* path) | | |
| 3参数 | MHandle hHYMRDHandle | | 库函数操作句柄 |
| const char \*path | | 读取文件路径 |
| 4 描述 | 读取描述符信息 | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 调用需要包含头文件"HYL\_MatchRigidBody.h"  需要包含库文件“MatchRigidBody.dll”“MatchRigidBody.lib” | | |
| 7 注意 |  | | |
| 8 举例 | #define MR\_READ\_FILE\_PARA "../MatchRigidBody.dat"  if (0 != HYL\_GetTemplateFromText(MHandle, MR\_READ\_FILE\_PARA))  {  printf("HYAMR\_GetTemplateFromText error !\n");  } | | |

## HYL\_GetDashboard

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 利用刚体匹配算法获取图像中目标对象的位置信息 | | |
| 2 语法 | 需包含头文件"HYL\_MatchRigidBody.h"  调用语法HYL\_GetDashboard (MHandle hHYMRDHandle,HYL\_PIMAGES pImage,MChar \*pClassName,MDouble threshold,PMPOINT offset) | | |
| 3参数 | MHandle hHYMRDHandle | | 库函数操作句柄 |
| HYL\_PIMAGES pImage | | 目标图像 |
| MChar \*pClassName | | 描述符类名 |
| MDouble threshold | | 刚体匹配阈值 |
| PMPOINT offset | | 物体中心点 |
| 4 描述 | 获取目标位置信息 | | |
| 5 返回值 | 0 | 接口运行正常 | |
| 非0 | 接口运行异常 | |
| 6 需求 | 调用需要包含头文件"HYL\_MatchRigidBody.h"  需要包含库文件“MatchRigidBody.dll”“MatchRigidBody.lib” | | |
| 7 注意 |  | | |
| 8 举例 | // pImg为调用opencv获取的图像  MPOINT \*centre;  HYL\_IMAGES src = {0};  centre = (MPOINT\*)malloc(1\*sizeof(MPOINT));  src.lHeight = pImg->height;  src.lWidth = pImg->width;  src.lPixelArrayFormat = HYL\_IMAGE\_BGR;  src.pixelArray.chunky.pPixel = pImg->imageData;  src.pixelArray.chunky.lLineBytes = pImg->widthStep;  HYL\_GetDashboard(MHandle, &src, "OK", 0.45, centre); | | |

## HYL\_MatchUninit

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 释放内存 | | |
| 2 语法 | 需包含头文件"HYL\_MatchRigidBody.h"  调用语法HYL\_MatchUninit(MHandle HYEDHandle) | | |
| 3参数 | MHandle HYEDHandle | | 待释放内存句柄 |
| 4 描述 | 释放内存 | | |
| 5 返回值 |  |  | |
|  |  | |
| 6 需求 | 调用需要包含头文件"HYL\_MatchRigidBody.h"  需要包含库文件“MatchRigidBody.dll”“MatchRigidBody.lib” | | |
| 7 注意 |  | | |
| 8 举例 | HYL\_MatchUninit(MHandle); | | |

# 结构体说明

## 图像格式

### 算法库图像格式

|  |  |  |
| --- | --- | --- |
| 参数名称 | KR\_IMAGES, \*KR\_PIMAGES | |
| 参数定义 | typedef struct {  int lWidth; // Off-screen width  int lHeight; // Off-screen height  int lPixelArrayFormat; // Format of pixel array  union  {  struct  {  int lLineBytes;  void \*pPixel;  } chunky;  struct  {  int lLinebytesArray[4];  void \*pPixelArray[4];  } planar;  } pixelArray;  }KR\_IMAGES, \*KR\_PIMAGES; | |
| 参数成员 | lWidth | 图像宽度 |
| lHeight | 图像高度 |
| lPixelArrayFormat | 图像格式，如RGB，YUV等 |
| pixelArray | 图像数据存储形式 |
| chunky | 图像按chunky格式存储，lLineBytes指图像的行长，\*pPixel为像素指针 |
| planar | 图像按planar格式存储， lLinebytesArray[4]为图像的行长， \*pPixelArray[4]为图像的像素指针 |
| 注意事项 | 需要将读取的图像转换为该图像格式，为接口函数调用，彩图默认通道顺序为‘BGR’ | |
| 相关接口 | HYKR\_KnobRecog\_GPU | |

### 刚体匹配图像格式

|  |  |  |
| --- | --- | --- |
| 参数名称 | HYL\_IMAGES, \*HYL\_PIMAGES | |
| 参数定义 | typedef struct {  MLong lWidth; // Off-screen width  MLong lHeight; // Off-screen height  MLong lPixelArrayFormat; // Format of pixel array  union  {  struct  {  MLong lLineBytes;  MVoid \*pPixel;  } chunky;  struct  {  MLong lLinebytesArray[4];  MVoid \*pPixelArray[4];  } planar;  } pixelArray;  } HYL\_IMAGES, \*HYL\_PIMAGES; | |
| 参数成员 | lWidth | 图像宽度 |
| lHeight | 图像高度 |
| lPixelArrayFormat | 图像格式，如RGB，YUV等 |
| pixelArray | 图像数据存储形式 |
| chunky | 图像按chunky格式存储，lLineBytes指图像的行长，\*pPixel为像素指针 |
| planar | 图像按planar格式存储， lLinebytesArray[4]为图像的行长， \*pPixelArray[4]为图像的像素指针 |
| 注意事项 | 需要将读取的图像转换为该图像格式，为接口函数调用，彩图默认通道顺序为‘BGR’ | |
| 相关接口 | HYL\_TrainTemplateFromMask，HYL\_GetDashboard | |

## 输出结果的结构体

|  |  |  |
| --- | --- | --- |
| 参数名称 | HYKR\_RESULT\_LIST | |
| 参数定义 | typedef struct  {  HYKR\_RESULT \*pResult;  MKRECT \*LRArea;  MKRECT MatchArea;  int lMaxResultNum; //最大数目  int lRealResultNum; //实际数目  int lResultNum; //程序得到结果数目  int lAreaNum;  MKPOINT origin;  MKPOINT offset;  }HYKR\_RESULT\_LIST; | |
| 参数成员 | pResult | 算法输出的目标结果 |
| LRArea | 交互框选的目标位置 |
| MatchArea | 交互框选的刚体匹配位置 |
| lMaxResultNum | 图中目标个数最大值 |
| lRealResultNum | 实际交互框选的目标个数 |
| lAreaNum | 交互框选的区域个数 |
| origin | 刚体匹配基准图中目标中心点 |
| offset | 待测图中目标偏移 |
| 注意事项 |  | |
| 相关接口 | HYKR\_KnobRecog\_GPU | |

### 单个结果的结构体

|  |  |  |
| --- | --- | --- |
| 参数名称 | HYKR\_RESULT | |
| 参数定义 | typedef struct  {  int flag;  double dVal;///最后的值  MKRECT Target;///目标rectangle  }HYKR\_RESULT; | |
| 参数成员 | flag | 结果是否输出，1输出，0不输出 |
| dVal | 输出类别，0是左，1是左上2是上，3是右上，4是右，5是右下，6是下，7是左下，8是钥匙向左，9是钥匙向右，10是横，11是竖 |
| Target | 目标位置 |
| 注意事项 | 无 | |
| 相关接口 | 无 | |

### 区域的结构体

|  |  |  |
| --- | --- | --- |
| 参数名称 | MKRECT, \*PMKRECT | |
| 参数定义 | typedef struct  {  int left;  int top;  int right;  int bottom;  } MKRECT, \*PMKRECT; | |
| 参数成员 | left | 左上角横坐标 |
| top | 左上角纵坐标 |
| right | 右下角横坐标 |
| bottom | 右下角纵坐标 |
| 注意事项 | 无 | |
| 相关接口 | 无 | |

### 坐标的结构体

|  |  |  |
| --- | --- | --- |
| 参数名称 | MKPOINT, \*PMKPOINT | |
| 参数定义 | typedef struct  {  int x;  int y;  } MKPOINT, \*PMKPOINT; | |
| 参数成员 | x | 点的横坐标 |
| y | 点的纵坐标 |
| 注意事项 | 无 | |
| 相关接口 | 无 | |