细指针算法使用说明

目录

[1、概述 2](#_Toc23960)

[1.1 概述 2](#_Toc20817)

[1.2 参考域说明 2](#_Toc5516)

[1.2.1 API参考域 2](#_Toc12756)

[1.2.2 数据类型参考域 2](#_Toc3205)

[2、 API参考 2](#_Toc28295)

[2.1 MeterReadRecogV3 2](#_Toc12334)

[2.2 yoloinit 3](#_Toc14949)

[2.3 yolouninit 3](#_Toc23467)

[3结构体说明 4](#_Toc28618)

[3.1图像格式 4](#_Toc25735)

[3.2保存交互信息的结构体 4](#_Toc26670)

[3.2.1刻度的坐标 5](#_Toc6161)

[3.2.2环形弧交互参数说明 5](#_Toc30353)

[3.3结果输出 8](#_Toc29831)

[4算法DEMO 8](#_Toc17008)

1、概述

## 概述

细指针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指针式仪表读数的算法模块，用于指针较细的的情况。

## 参考域说明

### 1.2.1 API参考域

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 |

### 1.2.2 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 |

1. API参考

2.1 MeterReadRecogV3

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 获取仪表读数 | | |
| 2 语法 | 需包含头文件"HY\_MeterRecogV3.h"  调用语法 MeterReadRecogV3(const void \*handle, HYLPMRV3\_IMAGES src, MV3Para Meterdescrip,HYMR\_POINTERRESULT \*MeterResult) | | |
| 3参数 | const void\* handle | | 模型空间句柄 |
| HYLPMRV3\_IMAGES src | | 待识别的仪表图像 |
| MV3Para Meterdescrip | | 保存交互信息的结构体 |
| HYMR\_POINTERRESULT \*MeterResult | | 读数结果 |
| 4 描述 | 获取仪表读数 | | |
| 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"HY\_MeterRecogV3.h"  需要包含库文件“yolo\_cpp\_dll\_no\_gpu.dll”“yolo\_cpp\_dll\_no\_gpu.lib”  “meterRecogV3.dll”“meterRecogV3.lib”  “halcon.dll”“halconcpp.dll”“halconxl.dll” | | |
| 7 注意 | 结构体定义见下面 | | |
| 8 举例 | MeterReadRecogV3(handle,&image, \*pOutPattern,MeterResult); | | |

2.2 yoloinit

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 初始化空间，加载模型 | | |
| 2 语法 | 需包含头文件"HY\_MeterRecogV3.h"  调用语法 yoloinit(void \*\*handle, char \*cfgfile, char \*weightfile, float Thresh) | | |
| 3参数 | void \*\*handle | | 模型空间句柄 |
| char \*cfgfile | | cfg文件路径（网络结构） |
| char \*weightfile | | weight文件路径（权值） |
| float Thresh | | 阈值，置信度 |
| 4 描述 | 初始化，加载模型 | | |
| 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"HY\_MeterRecogV3.h"  需要包含库文件“yolo\_cpp\_dll\_no\_gpu.dll”“yolo\_cpp\_dll\_no\_gpu.lib”  “meterRecogV3.dll”“meterRecogV3.lib”  “halcon.dll”“halconcpp.dll”“halconxl.dll” | | |
| 7 注意 | 使用后要使用对应的函数释放空间，无论返回值0还是非0 | | |
|  |  | | |

2.3 yolouninit

|  |  |  |  |
| --- | --- | --- | --- |
| 参考域 | 含义 | | |
| 1 目的 | 释放初始化的模型空间 | | |
| 2 语法 | 需包含头文件"HY\_MeterRecogV3.h"  调用语法 yolouninit(void \*handle) | | |
| 3参数 | void \*\*handle | | 模型空间句柄 |
| 4 描述 | 释放初始化的模型空间 | | |
| 5 返回值 |  |  | |
|  |  | |
| 6 需求 | 调用需要包含头文件"HY\_MeterRecogV3.h"  需要包含库文件“yolo\_cpp\_dll\_no\_gpu.dll”“yolo\_cpp\_dll\_no\_gpu.lib”  “meterRecogV3.dll”“meterRecogV3.lib”  “halcon.dll”“halconcpp.dll”“halconxl.dll” | | |
| 7 注意 |  | | |
|  |  | | |

3结构体说明

## 3.1图像格式

|  |  |  |
| --- | --- | --- |
| 参数名称 | HYMRV3\_IMAGES, \*HYLPMRV3\_IMAGES | |
| 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;  } HYMRV3\_IMAGES, \*HYLPMRV3\_IMAGES; | |
| 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 | MeterReadRecogV3 | |

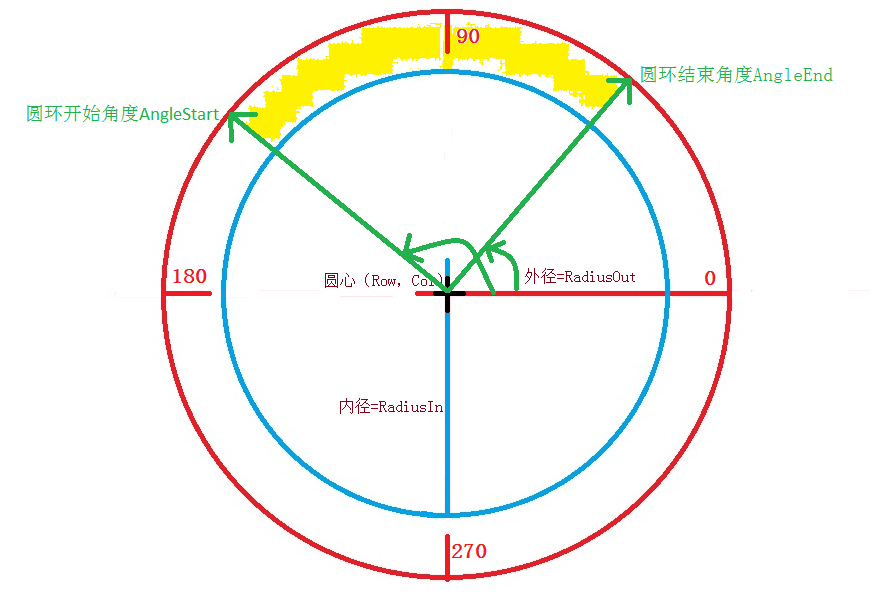
## 3.2保存交互信息的结构体

|  |  |  |
| --- | --- | --- |
| 参数名称 | MV3Para | |
| 参数定义 | #define HYAMR\_MAX\_PT\_LIST 30  typedef struct  {  long lPtNum;  MV3POINT ptPosList[HYAMR\_MAX\_PT\_LIST];  double dPtValList[HYAMR\_MAX\_PT\_LIST];  int AngleStart;  int AngleEnd;  int RadiusOut;  int RadiusIn;  int Row;  int Col;  }MV3Para; | |
| 参数成员 | cfgfile | cfg文件路径 |
| weightfile | Weights文件路径 |
| lPtNum | 刻度的个数 |
| ptPosList | 刻度的坐标 |
| dPtValList | 刻度的刻度值 |
| AngleStart | 环形弧开始的角度 |
| AngleEnd | 环形弧结束的角度 |
| RadiusOut | 环形弧的外径 |
| RadiusIn | 环形弧的内径 |
| Row | 圆心的横坐标 |
| Col | 圆心的纵坐标 |
| 注意事项 | 关于环形弧的交互信息 | |
| 相关接口 | MeterReadRecogV3 | |

3.2.1刻度的坐标

|  |  |  |
| --- | --- | --- |
| 参数名称 | MV3POINT | |
| 参数定义 | typedef struct  {  long x;  long y;  } MV3POINT, \*PMV3POINT; | |
| 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 | |

3.2.2环形弧交互参数说明



逆时针方向角度为正，顺时针方向角度为负。x轴正方向为0度，如上图所示。

上图黄色填充的环形弧区域为目标区域（指针刻度区域）。刻度点的位置也要标定在目标区域内，不能超出目标区域。

3.2.2.1示例图片

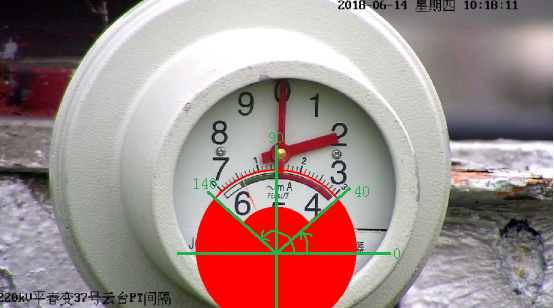


3.2.2.2对环形弧的交互数据可视化处理

环形弧参数：Col = 451，Row =407，RadiusOut = 130，RadiusIn = 70，AngleStart = 140，AngleEnd = 40



3.2.2.3逆时针角度（正）



程序内部会将环形弧状的指针刻度区域拉直为矩形。矩形为圆环从140变到40方向扫过的图像，如下图



如果将两角度交换，即AngleStart = 40，AngleEnd = 140时，矩形为圆环从40变到140方向扫过的图像，如下图所示



3.2.2.4顺时针角度（负）



此处AngleStart 逆时针角度为140，AngleEnd 处逆时针角度为40，取它们的顺时针角度AngleStart =-220，AngleEnd =-320。矩形为圆环从-220变到-320方向扫过的图像，与AngleStart = 140，AngleEnd = 40效果一样。



将两顺时针的角度交换，即AngleStart = -320，AngleEnd = -220时，矩形为圆环从 -320变到-220方向扫过的图像，如下图所示



## 3.3结果输出

|  |  |  |
| --- | --- | --- |
| 参数名称 | HYMR\_POINTERRESULT | |
| 参数定义 | typedef struct  {  double MeterValue;  }HYMR\_POINTERRESULT; | |
| 参数成员 | MeterValue | 仪表读数 |
| 注意事项 | 无 | |
| 相关接口 | MeterReadRecogV3 | |

4算法DEMO

|  |
| --- |
| #include<iostream>  #include "HY\_MeterRecogV3.h"  #include <windows.h>  #include <cmath>  #pragma comment(lib, "../Release/meterRecogV3.lib")  int train(const char \*input, MV3Para \*pOutPattern);  int detect(const char \*input, MV3Para \*pOutPattern);  int main()  {  while (1)  {  int count =0;  MV3Para para = { 0 };  //const char \*input = "D:/图片/tmp.bmp";  char single\_picname[100];//单张图片文件路径  printf("请输入单张照片的绝对路径,输入1退出\n");  scanf("%s", &single\_picname);  if (atoi(single\_picname) == 1)  break;  const char \*input = single\_picname;  train(input, &para);    detect(input, &para);    }  return 0;  }  int train(const char \*input, MV3Para \*pOutPattern)  {  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*交互操作获得圆环相关参数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*  pOutPattern->Col //圆心的横坐标  pOutPattern->Row //圆心的纵坐标  pOutPattern->RadiusOut //环形弧的外径  pOutPattern->RadiusIn //环形弧的内径  pOutPattern->AngleStart //环形弧开始的角度  pOutPattern->AngleEnd //环形弧结束的角度  \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*交互操作获得刻度相关参数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\*  pOutPattern->lPtNum //刻度的个数  pOutPattern->dPtValList[pOutPattern->lPtNum]//刻度的刻度值  pOutPattern->ptPosList[pOutPattern->lPtNum]//刻度的坐标  \*/  }  #define GrayStep(x) (( (x) + 3 ) & (~3) )  int detect(const char \*input, MV3Para \*pOutPattern)  {  char \*cfgfile = "../zhizhen/tiny-yolo-voc.cfg"; //模型路径  char \*weightfile = "../zhizhen/tiny-yolo-voc\_final.weights";//模型路径  IplImage \*src = cvLoadImage(input);  HYMRV3\_IMAGES image = { 0 };  HYMR\_POINTERRESULT MeterResult[1]={0};  image.lWidth = src->width;  image.lHeight = src->height;  image.pixelArray.chunky.lLineBytes = src->widthStep;  image.pixelArray.chunky.pPixel = src->imageData;  image.lPixelArrayFormat = HYAMR\_IMAGE\_BGR;  pOutPattern->cfgfile=cfgfile;  pOutPattern->weightfile=weightfile;  if(0!=MeterReadRecogV3(&image, \*pOutPattern,MeterResult))  {  printf("MeterReadRecogV3 failed!\n");  cvReleaseImage(&src);  return -1;  }  printf("MeterValue=%f\n",MeterResult[0].MeterValue);  cvReleaseImage(&src);  return 0;  } |

注：1、DEMO代码详见biaoji\_testbed.cpp。

2、biaoji\_testbed.cpp中train函数相关代码仅供参考，交互获得的环形弧的角度是错误的（由于cv::createTrackbar函数默认起始值为0，且不可更改，取值不能直接为负，画圆弧的函数cv::ellipse中设置圆弧角度的规则为顺时针方向角度为正，逆时针方向角度为负，x轴正方向为0度，与 3.2.2环形弧交互参数说明 中的角度规则不同，且生成扇形会遮盖目标区域，要通过取反操作来可视化目标区域），交互直接生成的角度仅用于可视化，不可直接使用。交互生成的其它的环形弧参数是对的。