**目录**

[虹软筛图总流程 3](#_Toc533854272)

[1、 原理介绍 3](#_Toc533854273)

[2、 总的流程图 3](#_Toc533854274)

[第二节：批量读取文件 4](#_Toc533854275)

[1、 功能 4](#_Toc533854276)

[2、 流程图 4](#_Toc533854277)

[3、 代码 4](#_Toc533854278)

[4、 接口描述 5](#_Toc533854279)

[第三节：获取图片特征 6](#_Toc533854280)

[1、 功能 6](#_Toc533854281)

[2、 流程图 6](#_Toc533854282)

[3、 代码 7](#_Toc533854283)

[4、 接口描述 8](#_Toc533854284)

[第四节：获取图片对比得分 8](#_Toc533854285)

[1、 功能 8](#_Toc533854286)

[2、 代码 8](#_Toc533854287)

[3、 接口描述 10](#_Toc533854288)

虹软SDK筛图程序

杭州图铭科技有限公司

2018.12

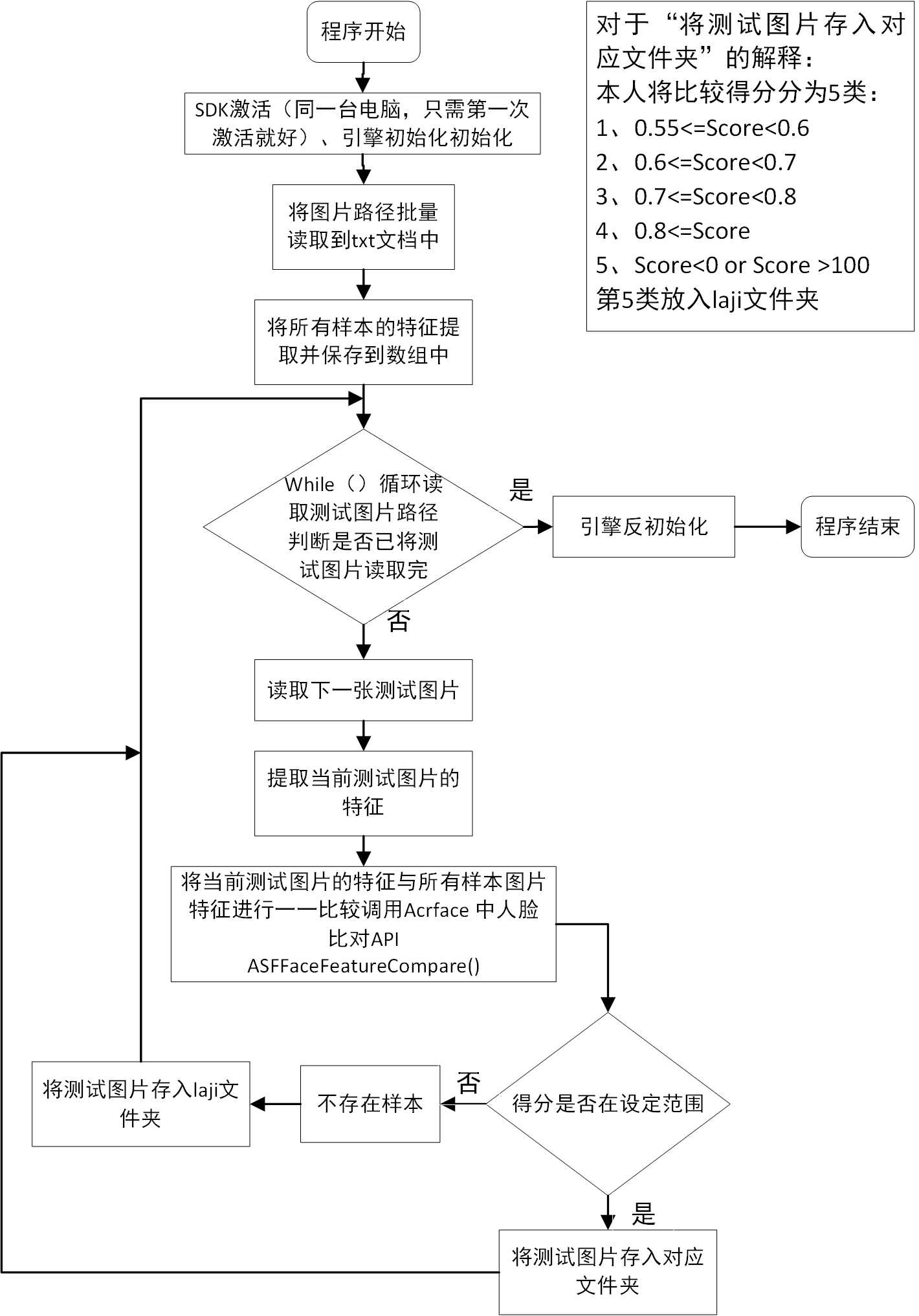
# 虹软筛图总流程

1. 原理介绍

**利用虹软（中国）开发的**[**Arcface v2.0 Windows（X64）C/C++ SDK**](https://ai.arcsoft.com.cn/ucenter/resource/openPlatform/index.html?r=0.8409129858642879)**进行人脸比对，以获得图片之间的比较得分。将图片分为为测试图片（待筛选）和样本图片（底库），按照比较得分分值的大小，将测试图片剪切到以自己命名的文件夹中，以实现图片分类的效果。**

**实现程序在链接中：**[**https://github.com/xuaikun/ArcsoftSDK\_FaceMatch**](https://github.com/xuaikun/ArcsoftSDK_FaceMatch)

1. 总的流程图



# 第二节：批量读取文件

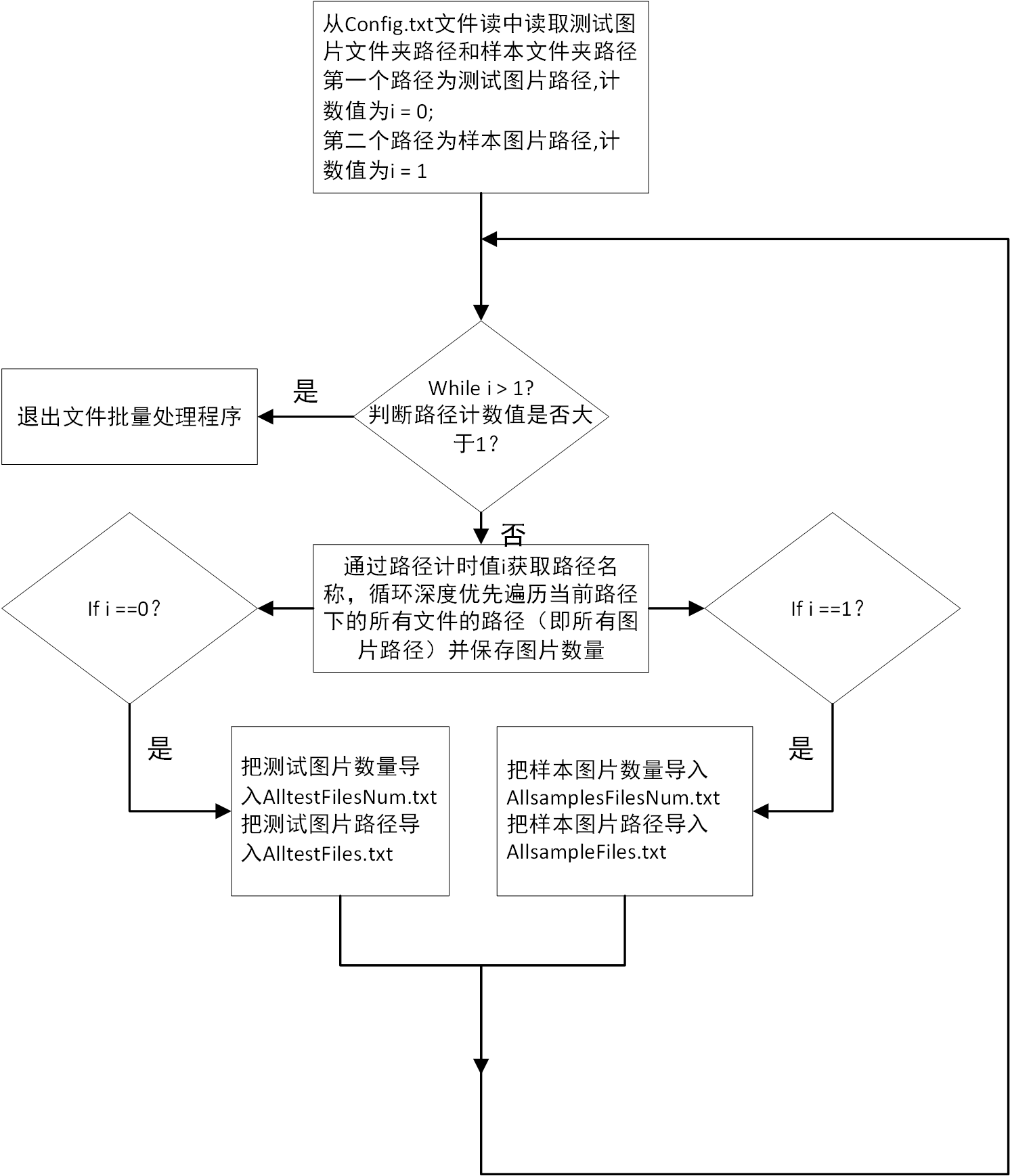
1. 功能

**从txt文件读取需要操作的文件夹的名称，包括测试图片文件夹路径、样本图片文件夹路径；深度优先（dfsFolder）遍历所有图片路径（读取到每个文件夹的最深处，即为图片路径名称），将图片数量和路径名称分别依次保存入不同文件（后期程序操作时，可以方便读取当前文件夹中图片数量和图片路径）。**

**实现代码链接如：**

[**https://github.com/xuaikun/ArcsoftSDK\_FaceMatch/blob/master/opencv\_test/ProcessFolder.cpp**](https://github.com/xuaikun/ArcsoftSDK_FaceMatch/blob/master/opencv_test/ProcessFolder.cpp)

1. 流程图



1. 代码

**深度优先遍历图片代码：**

**void dfsFolder(string folderPath, int i) {**

**\_finddata\_t FileInfo;**

**string strfind = folderPath + "\\\*";**

**intptr\_t Handle = \_findfirst(strfind.c\_str(), &FileInfo);**

**if (Handle == -1L)**

**{**

**cerr << "can not match the folder path" << endl;**

**exit(-1);**

**}**

**do {**

**//判断是否有子目录--**

**if (FileInfo.attrib & \_A\_SUBDIR) {**

**// cout<<FileInfo.name<<" "<<FileInfo.attrib<<endl;**

**//这个语句很重要--**

**if ((strcmp(FileInfo.name, ".") != 0) && (strcmp(FileInfo.name, "..") != 0)) {**

**string newPath = folderPath + "\\" + FileInfo.name;**

**//cout << FileInfo.name << " " << newPath << endl;**

**//根目录下下的子目录名字就是label名，如果没有子目录则其为根目录下**

**labelTemp = atoi(FileInfo.name);**

**// printf("%d\n",labelTemp);**

**dfsFolder(newPath, i);**

**}**

**}**

**else {**

**string finalName = folderPath + "\\" + FileInfo.name;**

**string finalName\_new = FileInfo.name;**

**//将所有的文件名写入一个txt文件--**

**// cout << FileInfo.name << "\t";**

**// printf("%d\t",label);**

**// cout << folderPath << "\\" << FileInfo.name << " " <<endl;**

**//将文件名字和label名字（子目录名字赋值给向量）--**

**//cout << "finalName\_new = " << finalName\_new << endl;**

**imgNames[i].push\_back(finalName);**

**imgNames\_real[i].push\_back(finalName\_new);**

**}**

**} while (\_findnext(Handle, &FileInfo) == 0);**

**\_findclose(Handle);**

**}**

1. 接口描述

**深度优先遍历图片代码接口**

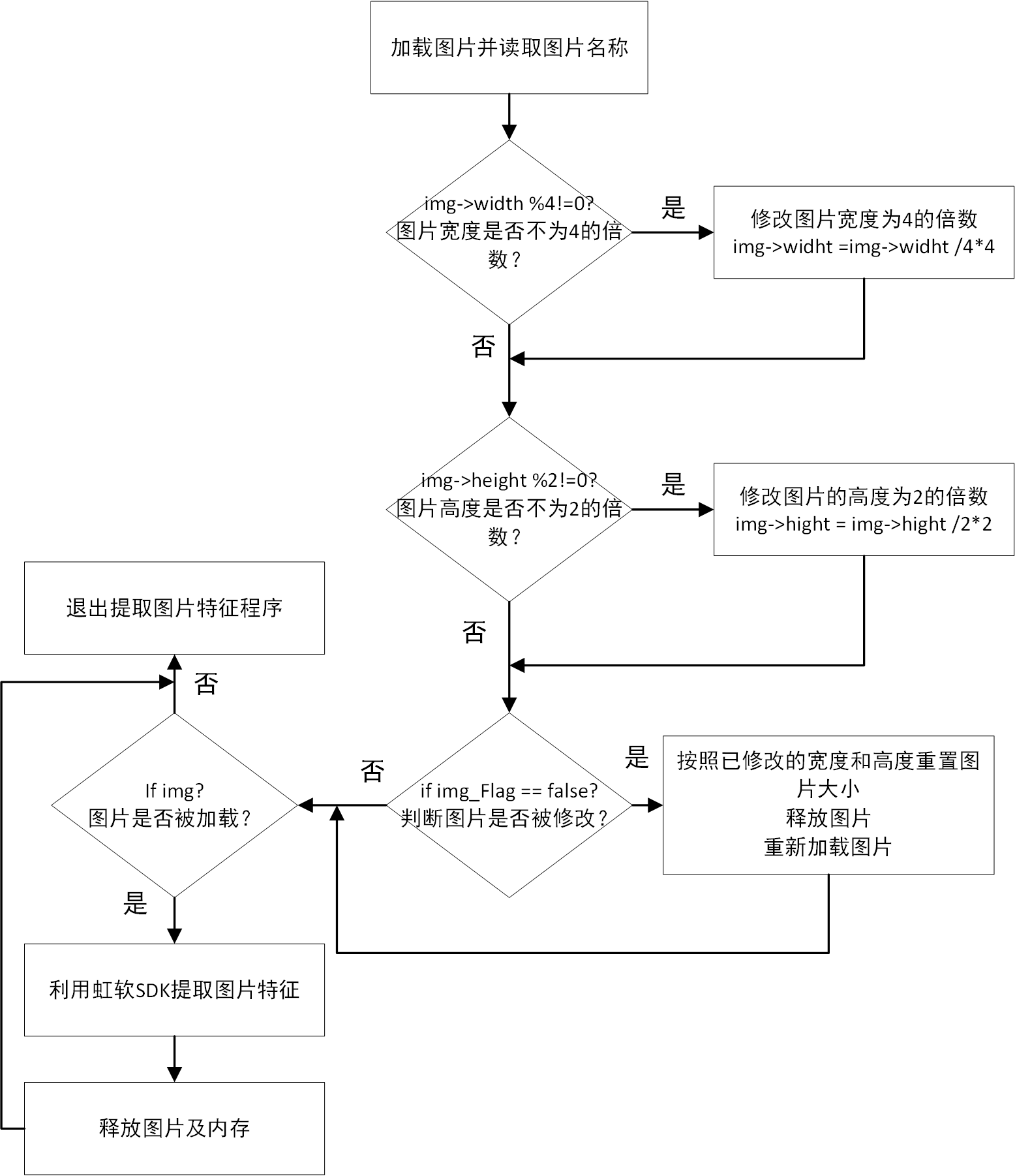
|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **In/out/Par** | **Size/precision** | **Description/value** |
| **folderPath** | **in** | **string** | **输入需要操作的文件夹路径** |
| **i** | **in** | **int** | **输入要操作的文件夹计数值** |

# 第三节：获取图片特征

1. 功能

**提取需要操作的图片的特征，以便后期程序操作使用，加快程序运行速度。代码链接如：**[**https://github.com/xuaikun/ArcsoftSDK\_FaceMatch/blob/master/opencv\_test/GetScore.cpp**](https://github.com/xuaikun/ArcsoftSDK_FaceMatch/blob/master/opencv_test/GetScore.cpp)

1. 流程图



1. 代码

**void feature\_obtain(MHandle handle, IplImage\* img, char\* imgname, LPASF\_FaceFeature my\_feature)**

**{**

**MRESULT res;**

**bool img\_Flag = true;**

**if (img->width % 4 != 4)**

**{**

**img->width = img->width / 4 \* 4;**

**img\_Flag = false;**

**}**

**//当高度不为2的倍数时，处理**

**if (img->height % 2 != 0)**

**{**

**img->height = img->height / 2 \* 2;**

**img\_Flag = false;**

**}**

**//cout << "img->width = "<< img->width << endl;**

**//cout << "img->height= " << img->height << endl;**

**if (img\_Flag == false)**

**{**

**Mat img\_new = imread(imgname);**

**resize(img\_new, img\_new, Size(img->width, img->height), 0, 0, CV\_INTER\_LINEAR);**

**imwrite(imgname, img\_new);**

**// 我觉得啊 到这里修改了一下，先把上一次的给释放了，然后再申请空间**

**cvReleaseImage(&img);**

**img = cvLoadImage(imgname);**

**}**

**if (img)**

**{**

**ASF\_MultiFaceInfo detectedFaces1 = { 0 };**

**ASF\_SingleFaceInfo SingleDetectedFaces1 = { 0 };**

**ASF\_FaceFeature feature1 = { 0 };**

**ASF\_FaceFeature copyfeature1 = { 0 };**

**res = ASFDetectFaces(handle, img->width, img->height, ASVL\_PAF\_RGB24\_B8G8R8, (MUInt8\*)img->imageData, &detectedFaces1);**

**if (MOK == res)**

**{**

**SingleDetectedFaces1.faceRect.left = detectedFaces1.faceRect[0].left;**

**SingleDetectedFaces1.faceRect.top = detectedFaces1.faceRect[0].top;**

**SingleDetectedFaces1.faceRect.right = detectedFaces1.faceRect[0].right;**

**SingleDetectedFaces1.faceRect.bottom = detectedFaces1.faceRect[0].bottom;**

**SingleDetectedFaces1.faceOrient = detectedFaces1.faceOrient[0];**

**res = ASFFaceFeatureExtract(handle, img->width, img->height, ASVL\_PAF\_RGB24\_B8G8R8, (MUInt8\*)img->imageData, &SingleDetectedFaces1, &feature1);**

**if (res == MOK)**

**{**

**//拷贝feature**

**copyfeature1.featureSize = feature1.featureSize;**

**copyfeature1.feature = (MByte \*)malloc(feature1.featureSize);**

**memset(copyfeature1.feature, 0, feature1.featureSize);**

**memcpy(copyfeature1.feature, feature1.feature, feature1.featureSize);**

**}**

**//else**

**//printf("ASFFaceFeatureExtract 1 fail: %d\n", res);**

**}**

**//else**

**//printf("ASFFaceFeatureExtract 1 fail: %d\n", res);**

**my\_feature->featureSize = copyfeature1.featureSize;**

**my\_feature->feature = (MByte \*)malloc(copyfeature1.featureSize);**

**memset(my\_feature->feature, 0, copyfeature1.featureSize);**

**memcpy(my\_feature->feature, copyfeature1.feature, copyfeature1.featureSize);**

**//cout << "my\_feature.featureSize =" << my\_feature.featureSize << endl;**

**SafeFree(copyfeature1.feature); //释放内存**

**}**

**cvReleaseImage(&img);**

**}**

1. 接口描述

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **In/out/Par** | **Size/precision** | **Description/value** |
| **handle** | **in** | **MHandle** | **引擎** |
| **img** | **in** | **IplImage** | **需要加载的图片** |
| **imgname** | **in** | **char\*** | **图片名称** |
| **my\_feature** | **out** | **LPASF\_FaceFeature** | **返回的图片特征** |

# 第四节：获取图片对比得分

1. 功能

**返回两张图片比较得分,程序链接如下：**

**<https://github.com/xuaikun/ArcsoftSDK_FaceMatch/blob/master/opencv_test/opencv_test.cpp>**

1. 代码

**res = ASFFaceFeatureCompare(handle, &sample\_feature[i], &test\_feature, &confidenceLevel);**

**根据得分将测试图片剪切到对应文件夹**

if (res != MOK)

{

// 表示图片异常

printf("ASFFaceFeatureCompare fail: %d\n", res);

Get\_Score = -1;

}

else

{

// 图片正常

Get\_Score = confidenceLevel;

}

if ((Get\_Score >= 55) && (Get\_Score < 60))

{

// 图片剪切到了指定文件夹，跳出循环

// insamplefile.close();

photo\_flag = true;

break;

}

else if ((Get\_Score >= 60) && (Get\_Score < 70))

{

// 将图片剪切到60\_70文件夹中

cout << strFinal << endl;

// 图片剪切到了指定文件夹，跳出循环

// insamplefile.close();

photo\_flag = true;

break;

}

else if ((Get\_Score >= 70) && (Get\_Score < 80))

{

// 将图片剪切到70\_80文件夹中

cout << strFinal << endl;

// 图片剪切到了指定文件夹，跳出循环

// insamplefile.close();

photo\_flag = true;

break;

}

else if (Get\_Score >= 80)

{

// 将图片剪切到80\_100文件夹中

cout << strFinal << endl;

// 图片剪切到了指定文件夹，跳出循环

// insamplefile.close();

photo\_flag = true;

break;

}

else if (Get\_Score < 0 || Get\_Score > 100)

{

//将这张图片剪切到垃圾文件夹中

cout << strFinal\_new << endl;

// insamplefile.close();

photo\_flag = true;

break;

}

1. 接口描述

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **In/out/Par** | **Size/precision** | **Description/value** |
| **handle** | **in** | **MHandle** | **引擎** |
| **&sample\_feature[i]** | **in** | **LPASF\_FaceFeature** | **样本图片特征** |
| **&test\_feature** | **in** | **LPASF\_FaceFeature** | **测试图片特征** |
| **&confidenceLevel** | **out** | **MFloat** | **返回的比较分值** |