

人脸识别算法SDK测试平台概要设计

V 1.0

智能安全技术研究中心

**版本更新**

| 日期 | 修订版本 | 修改章节 | 修改描述 | 作者 | 审核 |
| --- | --- | --- | --- | --- | --- |
| 2019/7/3 | V1.0 |  | 初始创建 | 刘勇 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目录

[1. 引言 1](#_Toc13214806)

[1.1 编写目的 1](#_Toc13214807)

[1.2 项目风险 1](#_Toc13214808)

[1.3 预期读者和阅读建议 1](#_Toc13214809)

[1.4 参考资料 1](#_Toc13214810)

[2. 设计概述 2](#_Toc13214811)

[2.1 开发环境 2](#_Toc13214812)

[2.2 限制和约束 2](#_Toc13214813)

[2.3 设计原则和设计要求 2](#_Toc13214814)

[3. 系统设计 2](#_Toc13214815)

[3.1 采用Googletest测试框架 2](#_Toc13214816)

[3.2 采用STL库完成平台内存管理 3](#_Toc13214817)

[3.3 Windows与Linux版本合一 3](#_Toc13214818)

[3.4 测试用例多种工作状态设计 4](#_Toc13214819)

[3.5 测试有效性设计 4](#_Toc13214820)

[3.5.1 接口覆设计 4](#_Toc13214821)

[3.5.2 错误码覆盖设计 5](#_Toc13214822)

[3.5.3 流程模拟测试 6](#_Toc13214823)

[3.5.4 分别测试单线程多线程性能 7](#_Toc13214824)

[3.5.5 对比测试单线程多线程提取结果 7](#_Toc13214825)

[3.6 测试自动化设计 8](#_Toc13214826)

[3.7 系统出错处理 8](#_Toc13214827)

[4. 数据库设计 8](#_Toc13214828)

[5. 待定问题 8](#_Toc13214829)

# 引言

## 编写目的

人脸识别算法SDK在开发人员自测的基础上，还必须设计一套覆盖方方面面场景，覆盖足够多的错误码、接口的测试用例，通过执行已设计测试用例保证版本质量。本自动化测试平台采用googletest框架，将大量测试用例有机组织在一起，通过自动化脚本的形式提供给测试人员，自动化执行设计测试用例，有效避免重复劳动，保障版本质量。

## 项目风险

无

## 预期读者和阅读建议

可能的读者包括：

* 开发人员；
* 项目经理；
* 测试人员；
* 文档编写人员；

## 参考资料

# 设计概述

## 开发环境

Windows 7 操作系统，VS2012编译器。

Ubuntu14.04操作系统，gcc 5.5.0编译环境。

## 限制和约束

无

## 设计原则和设计要求

设计原则：

1. 自动化测试，操作简单方便；
2. 正确性；
3. 有效性；

设计要求：

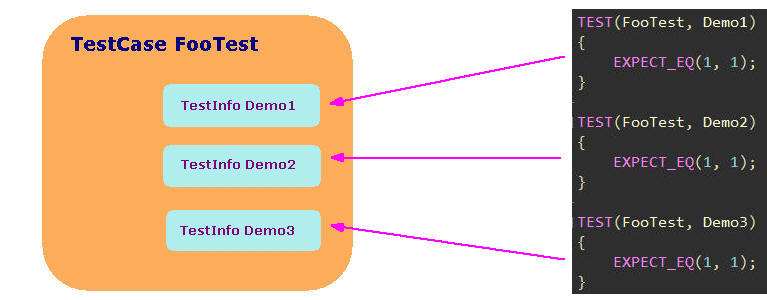
1. 支持Windows和Linux操作系统；
2. 支持单线程和多线程两种运行模式，单多线程测试结果一致性；
3. 测试SDK接口正确性、稳定性；
4. 按照人脸检测，特征提取，特征比对的流程组织测试；
5. 测试SDK内存消耗情况；
6. 测试人脸检测、特征提取成功率，特征比对正确率；
7. 测试接口耗时等。

# 系统设计

## 采用Googletest测试框架

Googletest测试框架  ，是在不同平台上（[Linux](https://baike.baidu.com/item/Linux/27050)，Mac OS X，Windows，[Cygwin](https://baike.baidu.com/item/Cygwin/151477)，Windows CE和[Symbian](https://baike.baidu.com/item/Symbian/264891)）为编写C++测试而生成的。

它是基于xUnit架构的测试框架，支持自动发现测试，丰富的断言集，用户定义的断言，death测试，致命与非致命的失败，类型参数化测试，各类运行测试的选项和XML的[测试报告](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95%E6%8A%A5%E5%91%8A)。



*图3-1 Googletest框架图*

## 采用STL库完成平台内存管理

STL（Standard Template Library），即标准模板库，是一个具有工业强度的，高效的C++程序库。它被容纳于C++标准程序库（C++ Standard Library）中，是ANSI/ISO C++标准中最新的也是极具革命性的一部分。

*char \*\*featureM = new char\*[numM];*

*for (int i=0;i<numM;i++)*

*{*

*featureM[i] = new char [8192];*

*}*

*图3-2 原有的成千次new操作*

测试平台采用STL库进行内存管理，之前版本中存在的数千次new操作全部消除，整篇代码没有一个new或者delete操作，有效降低测试平台内存管理出错的可能性。

## Windows与Linux版本合一

通过编译宏，实现测试平台Windows与Linux版本合一。

## 测试用例多种工作状态设计

在人脸检测测试用例中需要对X目录下的图片进行人脸检测处理；在特征比对时第一步也要进行人脸检测，测试用例一模一样只是图片文件在Y目录，要动态地让同一条测试用例根据需要去处理不同目录下的图片，需要在脚本中采用运行参数对可执行文件进行控制处理，如图3-2所示，增加了“smartirsec2018”控制参数。

测试平台可执行文件是多种测试用例的集合，在执行可执行文件时，可以根据需要，通过控制参数指定需要执行的测试用例，如图3-2所示，增加了“*--gtest\_filter=ftISDetTrack.ISFaceDetectRgb\_SingleThread*”控制参数。

*.\Debug\FaceSDKTest.exe* **smartirsec2018** *--gtest\_filter=ftISDetTrack.ISFaceDetectRgb\_SingleThread*

*图3-3 测试用例多种工作状态设计*

## 测试有效性设计

### 接口覆设计

按SDK提供的头文件，逐个接口遍历，能测试的都调用到。

|  |
| --- |
| .\include\SDK\ISFeature.h (22 hits) |
| int ISCreateFeatureChannel() |
| int ISCreateFeatureChannelEx(...) |
| int ISDestroyFeatureChannel(...) |
| int ISGetFeatureLength() |
| int ISGetFeature(...) |
| int ISGetFeatureEx(...) |
| int ISGetFeaturePath(...) |
| int ISGetFeaturePathEx(...) |
| int ISGetFeatureRgb(...) |
| int ISGetFeatureRgbEx(...) |
| int ISGetFeatureWithFacePos(...) |
| int ISGetFeatureWithFacePosPath(...) |
| int ISGetFeatureWithFacePosRgb(...) |
| int ISMergeFeature(...) |
| int ISGetPcaFea(...) |
| int ISpredictAgeGender(...) |
| int ISpredictGlasses(...) |
| int ISpredictSmile(...) |
| int ISpredictBeauty(...) |
| int ISpredictExpression(...) |
| int ISGetMaxFeatureChannel() |
| const char\* ISGetFeatureVersionInfo() |
| .\include\SDK\ISCompare.h (8 hits) |
| int ISCreateCompareChannel(); |
| int ISDestroyCompareChannel(...); |
| int ISCompare(...); |
| int ISCompareMN(...); |
| int ISCompareMNfasterprep(...); |
| int ISCompareMNfaster(...); |
| int ISGetMaxCompareChannel(); |
| const char\* ISGetCompareVersionInfo(); |
| .\include\SDK\ISDetTrack.h (17 hits) |
| int ISCreateDetTrackChannel(...); |
| int ISCreateDetTrackChannelEx(...); |
| int ISDestroyDetTrackChannel(...); |
| int ISFaceDetect(...); |
| int ISFaceDetectPath(...); |
| int ISFaceDetectRgb(...); |
| int ISFaceDetTrackRgb(...); |
| int ISCalFaceInfo(...); |
| int ISCalFaceInfo2(...); |
| int ISCalFaceInfoPath(...); |
| int ISCalFaceInfoPath2(...); |
| int ISCalFaceInfoRgb(...); |
| int ISCalFaceInfoRgb2(...); |
| int ISBlurQuaEvalRgb(...); |
| int ISCalFaceQualityRgb(...); |
| int ISGetMaxDetTrackChannel(); |
| const char\* ISGetDetTrackVersionInfo(); |

*表3-1 接口清单列表*

### 错误码覆盖设计

按SDK提供的文档，逐个制造错误场景，能触发的都调用到。

|  |  |
| --- | --- |
| .\include\SDK\ISError.h | |
| #define SUCC | 0 |
| #define UNKNOWN\_ERROR | 9999 |
| #define PARAM\_BUFFER\_TOO\_SMALL | 1001 |
| //图片相关 | |
| #define IMG\_DATA\_ERROR | 1100 |
| #define IMG\_SIZE\_TOO\_LARGE | 1101 |
| #define IMG\_SIZE\_TOO\_SMALL | 1102 |
| //算法相关 | |
| #define DETECTION\_ERROR | 1200 |
| #define DETECT\_NO\_FACE | 1201 |
| #define GET\_KEY\_POINT\_ERROR | 1202 |
| #define GET\_FEATURE\_ERROR | 1203 |
| #define COMPARE\_ERROR | 1205 |
| #define COMPARE\_MN\_ERROR | 1206 |
| #define TRACK\_ERROR | 1207 |
| //授权相关 | |
| #define UKEY\_CHECK\_ERROR | 1300 |
| #define NO\_DET\_TRACK\_AUTHORIZE | 1301 |
| #define NO\_FEATURE\_AUTHORIZE | 1302 |
| #define NO\_COMPARE\_AUTHORIZE | 1303 |
| #define DET\_CHANNEL\_NUM\_EXCEED | 1304 |
| #define FEA\_CHANNEL\_NUM\_EXCEED | 1305 |
| #define COMPARE\_CHANNEL\_NUM\_EXCEED | 1306 |

*表3-2 错误码覆盖率表*

### 流程模拟测试

按照接口的实用方式，依次进行人脸检测、特征提取、特征比对流程，计算每个过程性能信息等。

开始

人脸识别

识别结果存文件

识别结果读文件

特征提取

特征结果存文件

一比一特征比对

一比N特征比对

一比N快速特征比对

识别结果读文件

*图3-4 特征比对流程图*

### 分别测试单线程多线程性能

分别在单线程，多线程下进行测试，对比两种场景的性能指标。对于待测试接口：

第一，支持数据源位置目录可配置。

第二，与设备实用场景下接口调用方式一致，分A/B两个源目录进行提供，在config.ini文件中完成配置。

第三，单线程下接口净耗时，接口处理图片结果，内存消耗情况。

第四，多线程下接口净耗时，接口处理图片结果，内存消耗情况。

### 对比测试单线程多线程提取结果

分别在单线程，多线程下进行测试，对两种场景的运行结果进行一致性检查。

## 测试自动化设计

Windows环境采用批处理脚本调用可执行文件，在调用可执行文件时通过设定“--gtest\_filter”参数执行需要执行的测试用例，Linux采用python脚本实现，其它与Windows相同。

界面一：测试套列表

界面二：选定测试套下测试用例列表

动作一：执行选定测试用例

动作二：结束

*图3-5 自动化测试脚本架构*

具体参见《人脸识别算法SDK测试平台使用说明文档》。

## 系统出错处理

测试出错，平台会上报断言错误，保留现场，开发和测试平台人员一起定位错误原因。

# 数据库设计

无

# 待定问题

无