<Mardrobe智能穿衣搭配系统>

测试报告

版本 <2.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <12/7/2019> | <1.0> | <识别截取、测试> | <余张辉 > |
| <15/7/2019> | <1.1> | <特征提取测试> | <余张辉 > |
| <16/7/2019> | <1.2> | <连接数据库测试> | <高治远 > |
| <19/7/2019> | <1.3> | <图片上传返回测试> | <余张辉 > |
| <20/7/2019> | <2.0> | <搭配算法测试> | <高治远 > |

目录

1. 简介 4

1.1 目的 4

1.2 范围 4

1.3 定义、首字母缩写词和缩略语 4

1.4 参考资料 4

1.5 概述 4

2. 测试概要 4

3. 测试环境 4

4. 测试结果及分析 4

4.1 需求覆盖率及缺陷分布 4

4.2 缺陷严重程度 5

5. 缺陷清单 5

5.1 功能性缺陷 5

5.2 非功能性缺陷 6

6. 测试结论与建议 6

测试报告

# 简介

## 目的

测试第一次算法迭代产品的服装图片识别分类功能和搭配功能。

## 范围

相关项目：Mardrobe第一次算法部分迭代产品。

影响事物：后续前后端功能对接安排以及产品优化。

## 定义、首字母缩写词和缩略语

Mardrobe：Mardrobe智能穿衣搭配系统

## 参考资料

## 概述

针对第一次的算法迭代的简介，测试概要，测试环境与测试结果分析。

# 测试概要

时间：7.17-7.19

地点：软件学院3108

人员：余张辉 高治远

测试方法：手动测试

测试内容：服装截取识别测试 服装分类测试 服装搭配测试 功能测试

# 测试环境

软件：前端在 Webstorm node环境中测试图片读取写入；后端在Pycharm的虚拟环境中使用Tensorflow进行模型的运行

硬件：前后端均在Mac PC端运行

# 测试结果及分析

测试结果：后端模型之间衔接完善，各功能功能流畅运行，响应时间较短，搭配模型可进一步优化

结果分析：核心算法已完成，可进行小幅性能调优；搭配情况一般，缺少针对颜色进行的分类搭配，后续可优化这部分，提升实用性。

## 需求覆盖率及缺陷分布

**表1 需求覆盖率及缺陷分布**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试内容** | | **测试**  **用例数** | **需求**  **覆盖率** | **缺陷数** | **缺陷率** | **备注** |
| **功能项** | **识别截取** | 20 | 95% | 0 | 0 | 在图片中识别截取服装 |
| **特征提取** | 20 | 100% | 0 | 0 | 提取服装的特征并分类 |
| **搭配算法** | 20 | 80% | 0 | 0 | 通过算法找出完整搭配 |
| **连接数据库** | 20 | 100% | 0 | 0 | 通过模型写入数据库 |
| **图片上传返回** | 20 | 100% | 0 | 0 | 前端功能 图片-base64 |
| **非功**  **能项** | **性能** | 1 | 100% | 0 | 0 | 响应速度很快 |
|  |  |  |  |  |  |
| **…** |  |  |  |  |  |
|  | **非功能项小计** | **1** | **100%** | **0** | **0** |  |
|  | **总计** |  |  |  |  |  |

## 缺陷严重程度

**表2 缺陷严重程度**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **严重程度** | **致命** | **严重** | **一般** | **细微** | **总数** |
| **缺陷个数** | 0 | 0 | 0 | 1 | 1 |
| **占缺陷百分比** | 0 | 0 | 0 | 100% | 100% |

# 缺陷清单

## 功能性缺陷

**表3 功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **功能模块** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |

表中相关项说明：

1. 缺陷编号：为每个缺陷赋予一个唯一的编号，可以通过此编号对缺陷进行跟踪。例如：Bug001。
2. 缺陷严重程度：缺陷可以根据严重程度分为以下几种情况。
3. 致命（fatal）：致命的错误，测试执行直接导致系统死机、蓝屏、挂起、或是程序非法退出；系统的主要功能或需求没有实现。
4. 严重（critical）：严重错误，系统的次要功能点或需求点没有实现；数据丢失或损坏。执行软件主要功能的测试用例导致系统出错，程序无法正常继续执行；程序执行过于缓慢或是占用过大的系统资源。
5. 一般（major）：不太严重的错误，这样的缺陷虽然不影响系统的基本使用，但没有很好地实现功能，没有达到预期的效果。如次要功能丧失，界面错误，打印内容、格式错误，提示信息不太正确，或用户界面太差，简单的输入限制未放在前台进行控制，删除操作未给出提示，操作时间长等。
6. 细微（minor）：一些小问题，对功能几乎没有影响，产品及属性仍可使用。如软件的实际执行过程与需求有较小的差异；程序的提示信息描述容易使用户产生混淆；有个别错别字、 文字排列不整齐；辅助说明描述不清楚，显示格式不规范，长时间操作未给用户进度提示，提示窗口文字未采用行业术语，可输入区域和只读区域没有明显的区分标志，系统处理未优化等。
7. 功能模块：所测试并出现该缺陷的功能模块名称。
8. 缺陷标题：描述缺陷的标题。
9. 缺陷描述：对缺陷的详细描述。描述缺陷的重现步骤和问题所在，描述需清晰明了，使研发人员可以按照描述将缺陷重现出来。必要时可以放上相应的截图，以便于研发人员理解问题所在。

## 非功能性缺陷

**表4 非功能性缺陷列表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序**  **号** | **缺陷**  **编号** | **严重程度** | **测试类型** | **缺陷标题** | **缺陷描述** | **测试用例编号** |
| 1 | Color001 | 细微（minor） | 实用性 | 颜色识别条件 | 特征提取时未提取颜色，搭配时存在实用性问题 | 001 |
| 2 |  |  | 性能 |  |  |  |
| … |  |  | … |  |  |  |
| n |  |  | 可靠性 |  |  |  |

# 测试结论与建议

问题：特征提取模型效果一般，有待提升优化；未将颜色作为特征提取。

结论：基本功能完成度较高，性能表现很好，颜色会带来一些实用性影响。

建议措施：进一步优化性能，在特征提取模型环节加入颜色部分，提升实用性。（进阶功能）