计算机科学
COMPUTER SCIENCE

基于机器视觉的 Web 应用页面元素识别及可视化脚本生成

李子东，姚怡飞，王微微，赵瑞莲

引用本文

李子东，姚怡飞，王微微，赵瑞莲. 基于机器视觉的 Web 应用页面元素识别及可视化脚本生成[J]. 计算机科学, 2022, 49(11): 65-75.

LI Zi-dong, YAO Yi-fei, WANG Wei-wei, ZHAO Rui-lian. Web Application Page Element Recognition and Visual Script Generation Based on Machine Vision[J]. Computer Science, 2022, 49(11): 65-75.

相似文章推荐（请使用火狐或 IE 浏览器查看文章）

Similar articles recommended (Please use Firefox or IE to view the article)

基于多尺度多粒度特征的行人重识别

Multi-scale Multi-granularity Feature for Pedestrian Re-identification

计算机科学, 2021, 48(7): 238-244. https://doi.org/10.11896/jsjkx.200600043

图像处理在风电叶片法兰端面特征尺寸检测中的应用

Application of Image Processing in Feature Size Detection of Wind Turbine Blade’s Flange Face

计算机科学, 2019, 46(6A): 562-565.

基于日间行车的灯语识别技术

Lamp Language Recognition Technology Based on Daytime Driving

计算机科学, 2019, 46(11A): 277-282.

指针式仪表自动读数识别技术的研究现状与发展

Research Review and Development for Automatic Reading Recognition Technology of Pointer Instruments

计算机科学, 2018, 45(6A): 54-57.

基于 LabVIEW 机器视觉的餐具分拣系统

Tableware Sorting System Based on LabVIEW Machine Vision

计算机科学, 2018, 45(6A): 595-597.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220200122

基于机器视觉的 Web 应用页面元素识别及可视化脚本生成

李子东　姚怡飞　王微微　赵瑞莲
北京化工大学信息科学与技术学院　北京 100029
(lizidong0424@163.com)

摘　要　为了给用户提供丰富的交互响应,Web 应用的可视化元素越发复杂多样,传统基于 DOM 的测试已不能满足 Web 应用的测试新需求。新一代基于机器视觉的测试方法为 Web 应用复杂元素的测试提供了一种有效途径。目前,此类方法主要依赖于模版匹配技术识别 Web 页面元素,以生成可视化测试脚本对 Web 应用进行测试。然而,页面元素外观的细微变化可导致模版匹配技术失效,从而无法识别 Web 页面元素,更无法生成可视化测试脚本。因此,如何提高基于机器视觉的 Web 页面元素识别的准确性,使其在复杂条件中仍然适用是一项具有挑战性的工作。基于深度学习的目标检测是当前计算机视觉和机器学习领域的研究热点,可通过大样本学习得到深层的数据特征表示,以准确识别目标,其泛化能力相比模板匹配更强。针对 Web 应用,从页面元素的视觉特征出发,提出了一种基于深度学习的 Web 页面元素识别方法,即利用基于深度学习的目标检测算法 YOLOv3 构建 Web 页面元素识别模型,自动定位元素的位置和边界,识别 Web 页面元素类型及功能描述;在此基础上,自动为 Web 应用生成可视化测试脚本,提升 Web 应用的测试效率。为了验证基于机器视觉的 Web 页面元素识别的准确性,针对同一 Web 应用的不同版本及不同 Web 应用分别进行实验,结果表明,基于机器视觉的 Web 页面元素识别模型的平均召回率为 75.6%,可有效辅助 Web 应用可视化测试脚本生成。

关键词:Web 应用测试;Web 应用页面元素识别;可视化测试脚本;测试脚本自动生成;机器视觉
中图法分类号　TP391

Web Application Page Element Recognition and Visual Script Generation Based on Machine Vision

LI Zi-dong, YAO Yi-fei, WANG Wei-wei and ZHAO Rui-lian
School of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China

Abstract　In order to provide richer interactive response effect, the visualization elements of the Web application is becoming more complex and diverse. The traditional test based on DOM cannot match the new requirement to test Web application. A new generation test based on computer vision provides an efficient way for the complex elements in Web application. This test for the Web based on computer vision mainly depends on template matching technique to recognize the page elements, so that it can generate visualization test script. However, the subtle changes of the page elements' appearance can lead to the failure of template matching technique, so that the Web page elements cannot be recognized and the visualization test script cannot be generated. How to improve the accuracy of Web page element recognition based on machine vision and make it still applicable in complex conditions is a challenging task. Object detection based on deep learning is a research hotspot in the field of today's computer vision and machine learning. It has been shown from the deep data characteristics gained through the large sample learning that it can recognize the target accurately and has stronger generalization ability. Therefore, this paper targets the Web application, starts from the visual characteristics of the page elements, and proposes a Web page elements recognition method based on deep learning. This method uses YOLOv3 algorithmic structure model based on the target testing to automatically localize the position and boundary of an element, recognize the type of Web page elements as well as describe its function. On the base of the Web page elements recognition, it can automatically generate visualization test script for the Web application. To verify the accuracy of the page elements recognition based on computer vision, experiments are set to test between different versions of the same Web application, and between different Web applications to analyze its accuracy. The results show that the average recall rate of machine vision-based Web page element recognition model is 75.6%. It provides foundation for the Web application's visualization test script.

到稿日期:2022-02-21　返修日期:2022-05-26
基金项目:国家自然科学基金(62077003,61872026)
This work was supported by the National Natural Science Foundation of China(62077003,61872026).
通信作者:王微微(wangww@mail.buct.edu.cn)

Keywords Web application testing, Web application page element recognition, Visual test script, Automatic generation of test script, Machine vision

1 引言

随着 Web 技术的快速发展，Web 应用已经渗透到社会生活的各个领域。为了提升用户的交互体验，Web 应用的界面越来越复杂多样，传统基于 DOM 的测试方法已不再能满足 Web 应用测试的新需求，给 Web 应用软件测试带来了极大的挑战。基于机器视觉的测试方法为 Web 应用测试提供了一种新思路，该技术从 Web 应用的视觉特征出发，识别页面元素，生成可视化测试脚本，进而对 Web 应用进行测试。相比传统基于 DOM 的测试方法，基于机器视觉的 Web 应用测试可更轻松、直观地获取复杂的页面元素，实现测试脚本的生成。

目前，基于机器视觉的页面元素识别方法大多采用模板匹配技术定位识别元素，并利用自动化测试工具生成可视化测试脚本。例如，Sikuli[1]应用了 OpenCV 中归一化相关系数的模板匹配技术，将图案视为模板，并将模板与输入图像中等同大小的区域进行比较，以找到与模板最相似的区域，从而识别 Web 应用页面元素；VISTA[2]利用模板匹配技术中的尺度不变特征变换（Scale-invariant Feature Transform，SIFT）算法[3]，通过 SIFT 描述符从模板图像中提取关键点，并使用基于 Flann（Fast Library for Approximate Nearest Neighbors）的描述符匹配器进行关键点匹配，如果至少 70%的关键点被匹配，则执行快速归一化交叉相关模板匹配算法，识别 Web 应用页面元素。现有的模板匹配方法在解决 Web 页面元素识别问题上比较简单直观，但这些方法大多仅在像素级对比页面元素，泛化能力较弱，无法适应 Web 应用 UI 层面发生的变化，页面元素识别准确率较低。

基于深度学习的目标检测技术是当前计算机视觉和机器学习领域的研究热点，它利用卷积神经网络提取图像特征，识别图像中的目标，确定其位置与大小。该技术可有效适应 Web 应用 UI 层面的变化，泛化能力更强，适用于 Web 应用页面元素识别问题。YOLO[4]算法是一种常用的基于深度神经网络的实时目标检测算法，相比其他目标检测算法，它在拥有较快检测速度的同时保证了较高的精度，满足 Web 应用测试对动态性和实时性的需求。

因此，本文提出了一种基于机器视觉的 Web 页面元素识别方法，即利用深度学习中的目标检测 YOLO 算法，结合页面元素视觉特征，构建用于识别 Web 页面元素的 YOLO 模型，以识别页面中各个元素的类型及功能描述，并自动定位元素的位置和边界。在此基础上，针对测试人员的测试需求，自动构建 Web 应用可视化测试脚本。为了验证页面元素识别的准确性，选取云端软件开发协作平台 Gitee[5]中的 1 311 张各类 Web 应用页面作为实验数据集，构建 Web 应用页面元素识别模型，并针对同一 Web 应用的不同版本和不同 Web 应用进行实验，结果表明，当同一 Web 应用的不同版本之间的元素外观或页面布局发生变化时，本文方法可准确定位目标元素；再者，针对不同 Web 应用，本文方法仍适用，可有效支持可视化测试脚本的自动生成，提高 Web 应用的测试效率。

本文的主要贡献如下：

（1）提出了一种新的基于目标检测的 Web 应用页面元素识别方法，适用于同一 Web 应用的不同版本及不同 Web 应用的页面元素识别，相比现有基于 DOM 和基于模版匹配的元素识别方法，具有更加广泛的适用场景。

（2）提出了一种可视化测试脚本的自动生成方法，针对测试需求，结合页面元素及其功能描述，采用启发式搜索自动生成可视化脚本，以提升 Web 应用的测试效率。

（3）实验结果表明，本文提出的 Web 页面元素识别方法可针对同一 Web 应用视觉差异较大的两个版本，准确识别页面元素，也可针对不同的 Web 应用准确识别页面元素，其泛化能力相比现有方法有较大改善。同时，针对测试人员提出的测试需求描述，本文提出的可视化脚本生成方法可生成正确的测试脚本。

本文第 2 节介绍了目标检测算法 YOLO 与 Web 应用测试相关的技术；第 3 节介绍了本文基于机器视觉的 Web 页面元素识别及可视化脚本生成方法；第 4 节在同一 Web 应用的不同版本及不同 Web 应用两种场景中，分别使用本文提出的方法进行 Web 页面元素检测，分析其有效性，并进一步验证了可视化脚本生成的可靠性；第 5 节给出了相关研究，包括基于 DOM 的页面元素识别、基于模板匹配的页面元素识别；最后总结全文并展望未来。

2 基本概念及相关技术

2.1 目标检测算法 YOLO

YOLO（You Only Look Once）算法于 2016 年被 Redmon 等[4]提出，指只需浏览图片一次即可识别出图中的物体，确定其类别和位置。由于 YOLO 只需扫描一次图片，它又被称为 Region-free 方法，相比 Region-based 方法的代表算法 R-CNN[6]，YOLO 不需要提前找到可能存在目标的 Region，因此 YOLO 模型也被称为单阶段（One-stage）模型。YOLO 将目标检测问题定义为空间上分离的边界框和相关类别概率的回归问题，它无须使用复杂的计算框架，因此其检测速度较快，非常适合对实时性要求较高的任务。

此外，YOLO 算法基于整张图片信息进行预测，单个神经网络在一次评估中直接从完整图像预测边界框和类别概率，可以端到端训练以提高准确性。而其他滑窗检测框架只能基于局部图片信息进行推理。因此，YOLO 在拥有较快检测速度和较高准确率的同时，学到的图片特征也更为通用，当从自然图像推广到其他领域时，其明显优于其他方法[7]。

但是，YOLO 算法仍然存在一些缺陷。由于损失函数的

问题，定位误差是影响检测效果的主要原因，尤其是当一个格子中出现多于两个小目标或者一个小格中出现多个不同目标时，效果欠佳[8]。然而，这一缺陷在 Web 页面元素识别应用中并不会产生严重影响。一般而言，Web 应用为了保证整体页面的美观性，页面中的元素不会过于拥挤，这使得少许的定位识别偏差对最终的识别效果影响不大。即使出现一个元素识别框中截取少量其他不相关元素（识别框稍大）或者识别框截取元素完整性稍有欠缺（识别框稍小）的情况，均不会影响最终在此基础上的可视化脚本执行。

2.2 Web 应用测试相关技术

测试脚本的出现减轻了测试人员的工作负担，测试人员通过编写测试脚本并由自动化测试工具执行的方式，代替了人工手动执行大量重复操作的工作模式，大大提高了 Web 应用的测试效率。现有的 Web 应用测试脚本从其目标元素定位方式上可以分为基于 DOM 的测试脚本和基于视觉的可视化测试脚本。本节将分别介绍这两类 Web 应用测试脚本。

2.2.1 基于 DOM 的 Web 应用测试脚本

DOM 树描述了文档对象的逻辑结构，基于 DOM 的测试脚本通过定位并操作 DOM 树中的对象，来完成自动化测试工作[9-11]。其中，基于 DOM 的测试可以有多种方式定位元素，可以通过元素的 ID、元素的 name 属性和 XPath 语句来定位元素，以自动化测试工具 Selenium 的测试脚本为例，其以 ID 属性定位元素的方式为 find_element_by_id()。

在定位到元素之后，需要对元素进行点击、输入字符串等模拟操作。测试人员在编写 DOM 测试脚本的过程中，需要结合应用程序源代码才能正确定位并操作 Web 页面元素，当应用程序发生改变时，测试人员需要根据程序中发生的变化来修改维护 DOM 测试脚本。

2.2.2 基于视觉的 Web 应用可视化测试脚本

不同于 DOM 测试脚本，可视化测试脚本采用图片的方式定位元素，在运行的过程中采用图像识别功能识别元素。以 Sikuli 自动化测试工具编写的测试脚本为例，测试人员通过截取元素组件的截图，并调用 Sikuli 函数对其进行操作，形成测试脚本。图 1 给出了一个可视化测试脚本的例子。

图 1 基于 Sikuli 的可视化测试脚本

Fig. 1 Visual test script based on Sikuli

该脚本由 Sikuli 函数和元素截图组成，测试人员减少了与应用程序源代码的交互，相比 DOM 测试脚本大大降低了测试脚本的编写难度。当应用程序代码发生改变但 Web 应用的 UI 没有发生改变的情况下，可视化测试脚本不需要进行更改，降低了测试脚本的维护成本。

然而，Web 应用的 UI 变化可能会导致测试脚本无法使用，且在编写脚本的过程中需要测试人员对元素进行截图，这同样会消耗大量人力资源。因此，研究一种面向 Web 应用的可视化元素自动识别方法，对提高 Web 应用的测试效率至关重要。

3 基于 YOLO 算法的 Web 页面元素识别及可视化脚本生成

本文方法的整体框架图如图 2 所示，主要包含两个阶段：Web 页面元素识别与可视化脚本生成。Web 页面元素识别阶段旨在检测 Web 页面中每个可操作的页面元素，其中 Web 页面元素可由一个三元组〈类型，功能，位置〉表示。为了获取页面元素信息，本文利用 YOLO 算法构建 Web 应用页面元素识别模型，以确定页面元素的类型及位置；利用光学字符识别 OCR 技术识别元素自身及周围的文字信息；在此基础上，利用本文提出的图像文字融合算法得到元素的功能信息。可视化脚本生成旨在基于已识别的 Web 页面元素，利用启发式搜索算法生成满足测试需求的可视化测试脚本。

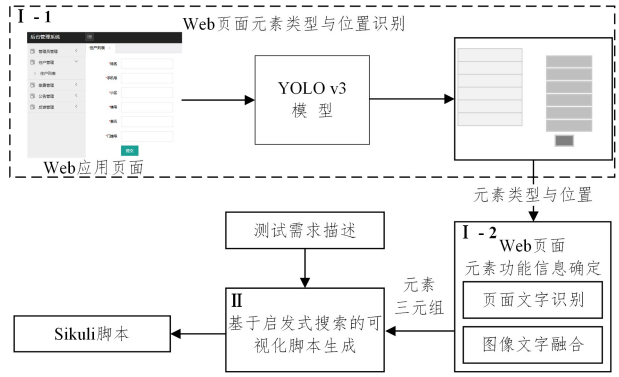

图 2 本文方法的整体框架

Fig. 2 Overall framework of the proposed method

3.1 Web 页面元素识别

Web 页面元素识别主要分为基于 YOLO 的 Web 页面元素类型与位置识别和 Web 页面元素功能信息确定这两个阶段。第一阶段根据页面元素外观，利用 YOLO 算法构建页面元素识别模型，得到页面元素的类型和位置信息。为了准确识别并定位 Web 页面元素，本文预先定义了 Web 页面涉及的 32 种元素类型，预定义元素类型如表 1 所列，定义规则将在 3.1.1 节中详细介绍。已知网页元素自身及周围的文字信息往往表征页面元素的功能，因此，为了确定元素功能，第二阶段首先利用 OCR 技术对网页文字进行识别，得到文字的内容和位置信息，然后利用图像文字融合算法，得到完整的页面元素三元组〈类型，功能，位置〉。

表 1 预定义元素类型

Table 1 Predefined element types

下拉框	输入框	表格	按钮	分页栏
搜索图标	日期选择	导航栏	标签页	多选框
单选框	日历	弹窗	开关	进度条
标签	通知图标	消息图标	折叠图标	添加图标
关闭图标	设置图标	分类图标	编辑图标	删除图标
上传图标	图表	返回图标	主页图标	退出图标
工具栏	刷新图标			

3.1.1 Web 页面元素类型定义

Web 页面中的元素复杂多样，每一种元素都有多种展现

形式，若不对各种元素进行详细的分类，则可能会造成元素类型混乱等问题。因此，本文首先根据元素控件的作用、位置特征、尺寸特征以及样式特征对控件进行预分类。观察发现，经过长期的发展，Web 应用页面从视觉上由一个个组件组成，如按钮、输入框、单选框和多选框等组件，只需要识别页面中原子粒度的元素（如图标或可组件化的元素，即按钮、输入框、多选框等）即可。

例如，对于 Web 页面中的多选框和按钮，其定义及标注规则如表 2 所列。

表 2 Web 页面中的多选框和按钮的定义及标注规则

Table 2 Definition and labeling rules of multiple checkboxes and buttons in Web pages

类型：多选框	作用：在一组备选项中进行多选
位置/样式特征：一个列表，可横向排列，也可垂直排列，常出现于表单中，可选择多项，每项头部为一个方框，选中后会变得与其他选项不同	
样例：☑多选框1 ☐多选框2 ☐多选框3	
标注规则：☑多选框1 ☐多选框2 ☐多选框3	
类型：按钮	作用：常用的操作按钮
位置/样式特征：有明显的轮廓特征，呈正规长方形或圆角长方形状，中间为一小段文字或图标，会有不同尺寸。常出现于表单中用于提交、删除等，可表现出多种颜色	
样例：	按钮 按钮
标注规则：	按钮 按钮

3.1.2 基于 YOLO 的 Web 页面元素类型与位置识别

Web 应用页面元素识别模型，本文利用 YOLO 算法训练大量 Web 页面数据来提取各类元素的视觉特征，并利用这些视觉特征确定 Web 元素的类型及位置。

具体而言，首先对收集的 Web 页面按照 3.1.1 节介绍的标注规则进行标注；然后，对输入的 Web 页面元素图像样本数据，使用借鉴了残差网络结构的 YOLOv3[7] 模型，通过一系列 3 * 3 和 1 * 1 的卷积操作来提取相应特征，进行组合学习，从而训练得到特征图模型；并在此基础上添加特征交互层，分为 3 个尺度，每个尺度内通过卷积核的方式实现特征图局部的特征交互，根据数据标准化处理以及维度聚类、细粒度特征操作，直接预测出 Web 页面元素的中心坐标；同时，引入惩罚机制来提高模型的泛化能力，以更好地匹配定位 Web 页面元素，在此基础上添加多标签多分类的逻辑回归层，实现对目标元素进行分类识别。详细的 YOLOv3 模型的训练过程将在 4.3 节介绍。借鉴残差网络结构的 YOLOv3 模型，解决了传统的深度学习中因卷积神经网络的层数增加而出现的梯度消失和梯度爆炸情况，从而使得 Web 应用页面元素识别模型具有较好的收敛性和鲁棒性。

3.1.3 Web 页面元素功能信息确定

（1）页面文字识别

Web 页面元素周围的文字往往对其功能起着提示作用，例如按钮中的文字可以代表当前按钮所能触发的事件和功能，输入框左侧的文字一般代表当前输入框需要输入的信息等。为了得到 Web 页面元素的功能信息，在确定目标元素的类型及位置后，需对元素自身及其周围的文字信息进行识别。OCR 技术可实时高效地定位并识别图片中的所有文字信息，进而得到文字框位置与文字内容[12]。但 Web 页面内的文字识别不同于传统的文字识别，页面中的文字有着大小不一、长短不一、紧凑程度不同等特点。因此，要求 OCR 接口有以下特点：精度高，能够去除背景的影响从而识别各种艺术字体，字段划分合理，同时还希望能够识别出文字的位置以及边界信息。综合考虑以上需求，本文选用百度 OCR 接口来识别并获取页面文字信息。

（2）图像文字融合

为了获取页面元素对应的功能信息，需将元素类型与位置信息和页面文本信息进行有机融合，将二者关联起来。已知在 Web 应用中，页面元素与周围文字通常有较大的关联，考虑到 Web 页面中的元素与文字位置关系通常存在以下关系，即左右关系和包含关系，本文的页面元素图像和文字的融合对应以下 3 种情况：1）元素＋左侧文字融合；2）元素＋右侧文字融合；3）元素＋元素内文字融合。

不同类型的页面元素一般对应着不同的元素与文字位置关系，相同类型的页面元素一般位置关系相同。因此，对于不同的元素类型，本文采用不同的融合方式，具体如下。

（1）下拉选择框：一定范围内向左扩展（可能有，也可能无），提取元素左边的文字作为元素的功能信息.

（2）输入框：一定范围内向左扩展（可能有，也可能无），提取元素左边的文字作为元素的功能信息。当输入框左边没有文字时，使用输入框内的文字作为功能信息。

（3）按钮：提取元素内的文字作为按钮的功能信息。

（4）多选框：一定范围内向右扩展，提取元素右边的文字作为元素的功能信息。

向左融合方法如算法 1 所示。针对每个由 YOLO 模型识别出的页面元素，遍历由 OCR 算法检测出的当前页面的文字列表，若存在一个位于页面元素左方的文字识别框的右边界与元素识别框的左边界之间的距离小于 10 个单位，且该框的上下边界位于元素的上下边界之间，则判定该文字与页面元素相关（见算法 1 中的第 3－5 行），然后从文字列表中删除位于页面元素内部的文字识别框以减少下次元素融合的计算量（见算法 1 中的第 6 行），继而将文字内容作为页面元素的功能信息保存，扩大元素识别框，使之包围该文字框（见算法 1 中的第 7－9 行），最后将该文字框从文字列表中删除并返回处理后的页面元素对象。若遍历当前所有文字后均未找到相关文字，则转为在页面元素内部寻找相关文字（见算法 1 中的第 14 行）。向右融合方法与上述方法类似，不再赘述。在得到完整的页面元素三元组＜类型，功能，位置＞后，对扩展后的标识框进行截图保存（见算法 1 中的第 23 行），并将文件命名为元素类型＋功能，为后续可视化测试脚本的生成奠定基础。

算法 1 Web 应用页面元素检测算法

Input：Screenshot of Web application page img

```
Output: Array of elements in the page ImgObj[]
1. function FUSIONLEFT(obj, Text[])
2.     for i=0; i < Text.size(); i++ do
3.         if 0 < obj.left − Text[i].left − Text[i].width < 10
4.             and Text[i].top >= obj.top
5.             and Test[i].top + Test[i].height < obj.top + obj.height
               then
6.             DELETEINNER(obj, Test)
7.             obj.width = obj.left + obj.width − Text[i].left
8.             obj.left = Text[i].left
9.             obj.fuction = Text[i].eleName
10.            Text.remove(i)
11.            return obj
12.        end if
13.    end for
14.    return SEARCHINNER(obj, Test)
15. end function
16. function ELEMENTDETECT(img)
17.    ImgObj[] = null
18.    Obj[] = YOLO(img)
19.    Text[] = OCR(img)
20.    for i=0; i < Obj.size(); i++ do
21.        element = FUSION(Obj[i], Text[])
22.        ImgObj.add(element)
23.        ELEMENTCROP(element)
24.    end for
25.    return ImgObj[]
26. end function
```

3.2 可视化脚本自动生成

在 Web 页面元素识别的基础上，本节将探讨一种可视化测试脚本生成方法，以自动生成满足测试人员需求的测试脚本。由 2.2.2 节可知，可视化测试脚本的编写需获取元素的截图及其类型作为参数。在可视化测试脚本生成过程中，本文针对测试人员的需求，将其与元素功能描述信息进行相似度匹配，以识别 Web 页面目标元素，获取其截图及类型信息，自动构建满足需求的可视化脚本。

测试人员的测试需求描述一般是以一组操作步骤的形式给出，每个步骤由动作、对象和输入值 3 部分组成，即〈动作，对象，输入值〉，其中输入值可为空。例如，〈输入，输入框：姓名，“admin”〉表示在姓名输入框中输入“admin”；〈点击，按钮：添加，“空”〉表示单击添加按钮。测试需求是对测试动作和目标对象元素的描述，而 Web 应用页面元素由三元组〈类型，功能，位置〉表示，因此本文以测试需求与页面元素类型及功能描述的匹配程度为启发式策略，搜索出相应的 Web 页面元素，生成测试脚本。

如算法 2 所示，具体而言，从 Web 应用首页开始，对于每一条测试需求描述，截取当前网页的图像（见算法 2 中的第 5 行），经 Web 应用页面元素识别处理后得到首页中每个页面元素的三元组及其对应的组件截图（见算法 2 中的第 6 行）；然后利用字符串匹配算法将测试需求对象属性和页面元素类型与功能属性进行相似度匹配（见算法 2 中的第 9 行），若存在一个页面元素与当前测试需求互相匹配，则将测试需求与页面元素作为参数，生成一条可运行的 Sikuli 脚本，然后触发对应的事件，进入一个新的页面中（见算法 2 中的第 15－18 行），再次截取整个网页截图进行 Web 页面元素识别；若匹配失败，则考虑当前页面尚未加载完成，截图时机过早会导致截图内没有匹配的页面元素，此时应等待 3 s 后再重新截图。循环执行上述步骤，直到所有的测试需求语句全部处理完毕。

算法 2 测试脚本自动生成

```
Input: Array of test requirement descriptions TRD[], website address
       url
Output: Array of visual test script TestScript[]
1. function TESTSCRIPTGENERATION(TRD[], url)
2.     TestScript[] = null
3.     OPEN(url)
4.     for i=0; i < TRD.size(); i++ do
5.         img = GETSCREENSHORT()
6.         ImgObj[] = ELEMENTDETECT(img)
7.         flag = 0
8.         for j=0; j < ImgObj.size(); j++ do
9.             if Match(TRD[i], ImgObj[j]) then
10.                flag = 1
11.                matchedElement = Imgobj[j]
12.                break
13.            end if
14.        end for
15.        if flag == 1 then
16.            script = TRANSFORM(TRD[i], matchedElement)
17.            TestScript.add(script)
18.            EXECUTE(script)
19.        else
20.            WAIT(3)
21.            i−−
22.        end if
23.    end for
24.    return TestScript[]
25. end function
```

4 实验结果与分析

为了验证基于机器视觉的 Web 页面元素识别方法的有效性，本文针对同一 Web 应用的不同版本及不同的 Web 应用两种场景，分别探讨了 Web 页面元素识别的准确性。进一步地，为了验证可视化脚本生成方法的可靠性，本文又对自动生成的测试脚本进行人工检验，判定其能否满足测试人员的测试需求。

4.1 实验数据集与环境

本文以 Gitee（云端软件开发协作平台）中搜集的 1 311 张各类 Web 应用页面截图（见表 3）及 3 种 Web 应用程序 Mrbs[13]，OrangeHRM[14]，MantisBT[15] 的两个不同版本，My-

Admin[16]、VUE-Manage-System[17]、GABS[18]、常春藤物业缴费系统等开源应用作为实验数据集。

表3 数据集中截取的各类Web应用的页面数量

Table3 Number of pages of various Web applications intercepted in dataset

网站名称	页面数	网站名称	页面数	网站名称	页面数
学无止境教育管理后台	6	妖气山视频管理	9	OFCMS	6
在线教育校务管理系统	38	L小说精品屋	10	XENON	7
学之思考试系统	7	B2不二掌柜	5	kitty Platfrom	4
学之思后台管理系统	4	知了CMS	8	NACOS	4
领课教育管理系统	14	TalkingData	3	Oasys	6
万岳教育开源管理系统	23	私人医疗健康管理平台	6	PB-CMS	8
Crmeb课程管理系统	29	智慧督办管理系统	10	管理云平台	4
云帆考试系统	11	政务督办管理系统	47	MACOS管理系统	5
教育后台管理	7	医院信息管理系统	13	电商管理后台	26
云教务	3	健康档案后台管理系统	11	电商后台-商品管理部分	1
EXAM++网络考试系统	4	自然资源和规划局	23	渠道商运营管理后台	47
码蚁成绩管理系统	2	律所后台系统	17	YSHOP电商系统	6
Tamguo探果课程	7	药物进货销售管理系统	10	供货商城系统	10
CD学习管理系统	2	HC小区管理系统	2	新蜂商城	6
员工薪资管理系统	11	智慧社区物业管理系统	49	新蜂商城管理系统	6
高保工单管理系统	8	智慧食堂进销存	6	mall商城管理系统	21
微人事	4	物资管理系统	33	Xmall	6
Saas人事管理系统	5	二手交易网站后台管理	9	优惠卷管理系统	9
人事管理系统	4	猫宁后台管理系统	11	聚商汇WMS	12
团餐数据中控	31	YZNCMS后台管理系统	4	Likeshop	7
若依用户管理系统	20	白卷后台管理系统	6	客户智慧营销平台	3
权限管理系统	5	Eladmin后台管理系统	7	Jeecg物流管理系统	8
住户管理系统	6	高保真资讯后台原型	7	通用运营后台	6
酒店管理系统	10	若依管理系统	13	会员积分管理后台	16
FEBS权限系统	10	综合管理平台系统	6	知启蒙代码仓库	8
企业ERP进销存管理后台	21	TIMO后台	7	码上无忧	4
企业网站管理后台	9	档案管理后台操作系统	47	梦想家内容发布系统	9
JFinal极速开发企业管理系统	8	V部落博客管理平台	5	文档在线协作管理平台	34
华夏ERP	11	码匠社区	3	数据分析系统	7
客运行业大数据监管平台	2	MarkHub博客系统	6	青阳网络文件传输系统	5
宝葫芦集团管理系统	5	巡云轻论坛	11	Skyeye网络桌面系统	9
同创ICS	16	Personal Blog	5	PayCould支付系统	15
Tamguo Oms	3	易知网	8	mall后台管理系统	6
财务报销管理后台	6	Sentinel控制管理系统	4	餐饮管理后台	12
ZSZ权限管理系统	3	Kubernetes监控系统	8	妖气山后台管理	5
设备资产管理	4	SpringBlade接口文档系统	4	YzmCMS内容管理系统	27
工单管理系统	16	lamp-cloud系统管理	3	Skyeye后台管理系统	22
TMS沟通	6	Smart OA框架	11	Fantastic-admin后台管理系统	6

基于机器视觉的Web应用页面元素识别的相关配置为NVIDIA GeForce RTX 3090型号GPU，训练模型为YOLOv3，深度学习框架为DarkNet，编程语言为Python。

可视化脚本的自动生成方法的实验环境为PC机，相关配置为MacBook Pro(16-inch,2019)，2.3GHz Intel Core i9处理器，16GB运行内存，编程语言为Python。

4.2 实验设计

实验将在两种不同场景中分别使用本文提出的基于机器视觉的Web页面元素识别方法，以验证元素识别的准确率，以及测试脚本的可靠性。本文提出了以下4个研究问题。

研究问题1：本文提出的Web页面元素识别方法在测试集中的识别效果如何？在同一Web应用GUI发生变化的不同版本中使用本文方法，是否能够准确识别出页面中元素的类型及其位置信息？

研究问题2：本文提出的Web页面元素识别方法对不同类型的Web应用是否有效？

研究问题3：本文提出的融合方法能否有效地将Web应用页面元素图像信息与功能描述关联起来？

研究问题4：根据测试需求描述生成的Sikuli测试脚本能否正确地完成测试？

针对以上研究问题，本文设计了4个实验分别进行分析。为了验证本文方法的Web页面元素识别效果，以元素类型识别的召回率和对元素位置边界的定位效果作为Web页面元素识别效果的评价指标。

4.3 Web页面元素识别模型的训练过程

本文的训练数据采集了Gitee(云端软件开发协作平台)中的1311张各类Web应用页面作为数据集，标注了包括按钮、输入框、单选框、多选框、表格等在内的32种常见元素类型，且使用随机抽取的方法选取训练集∶验证集∶测试集的比例为0.72∶0.18∶0.1。3种集合中各类元素的具体数目如表4—表6所列。选用轻量级深度学习框架Darknet作为训练框架，并选择借鉴ResNet残差网络结构的Darknet-53，以保证在网络层深较大的情况下仍能收敛。在训练过程中，我们将最大迭代次数(max_batches)设置为64000；在Darknet

中，批次(Batch)和细分(Subdivisions)是结合使用的，我们将 batch 设置为 64，subdivisions 设置为 16，表示训练的过程中将一次性加载 64 张图片进内存，然后分 16 次完成前向传播，即每次 4 张，前向传播的循环过程中累加 loss 求平均，待 64 张图片都完成前向传播后，再一次性后向传播更新参数。

表 4 训练集中各类元素的数量

Table 4 Number of various elements in training set

元素类型	数量	元素类型	数量
下拉选择框	715	进度条	52
输入框	1614	标签	52
按钮	3538	通知	284
表格	427	消息	116
分页框	263	折叠	431
搜索框	311	添加	97
日期选择框	122	关闭	259
导航栏	767	设置	46
刷新图标	2	分类	13
图表	61	编辑	214
标签页	133	删除	126
多选框	2046	上传	29
单选框	336	返回	24
日历	8	主页	14
弹框	113	退出	55
开关	232	工具栏	30

表 5 验证集中各类元素的数量

Table 5 Number of elements in verification set

元素类型	数量	元素类型	数量
下拉选择框	166	进度条	25
输入框	408	标签	12
按钮	883	通知	62
表格	115	消息	32
分页框	69	折叠	114
搜索框	94	添加	24
日期选择框	27	关闭	76
导航栏	189	设置	22
刷新图标	3	分类	1
图表	13	编辑	40
标签页	34	删除	24
多选框	712	上传	6
单选框	87	返回	7
日历	1	主页	6
弹框	31	退出	18
开关	88	工具栏	8

表 6 测试集中各类元素的数量

Table 6 Number of various elements in test set

元素类型	数量	元素类型	数量
下拉选择框	90	进度条	23
输入框	263	标签	5
按钮	470	通知	34
表格	76	消息	17
分页框	43	折叠	57
搜索框	43	添加	19
日期选择框	15	关闭	44
导航栏	111	设置	11
刷新图标	2	分类	2
图表	3	编辑	27
标签页	23	删除	16
多选框	302	上传	3
单选框	51	返回	5
日历	3	主页	3
弹框	14	退出	8
开关	16	工具栏	4

由于学习率决定着权值更新的速度，设置得过大会使结果超过最优值，过小会使下降速度过慢，本次实验将学习率(learning_rate)经验设置为 0.01，并采用逐步降低学习率的方法进行实验，对应的训练策略(Poilcy) 选择分步(Steps) 策略，当迭代次数达到 51 200 和 56 700 时分别衰减为原来的 1/10。优化方案采用动量法，该参数值影响着梯度下降到最优值的速度，本次实验设置为 0.9。为了防止过拟合，权重衰减正则项值设置为 0.000 5。由于本次实验的目的是识别 32 种常见的 Web 元素类型及位置，因此将最后一层卷积核数量 n 设置为$(32+5)\times 3=111$。

交并比(IOU)阈值(ignore_thresh)参数控制了参与 loss 计算的检测框的规模，如果 ignore_thresh 设置得过大，参与检测框回归 loss 的个数就会较少，同时也容易造成过拟合；而如果 ignore_thresh 设置得过于小，那么参与计算的数量规模就会很大，同时也容易在进行检测框回归时造成欠拟合，因此本文将 ignore_thresh 设置为 0.7。当预测的检测框与 ground true 的 IOU 大于 0.7 时，参与 loss 的计算，否则不参与 loss 计算；同时，为了提高目标检测的正确率，本文选择打开随机多尺度训练，width 会在 320608 之间随机取值，且 width=height，每 10 轮随机改变一次。训练过程中的 loss 值变化如图 3 所示。

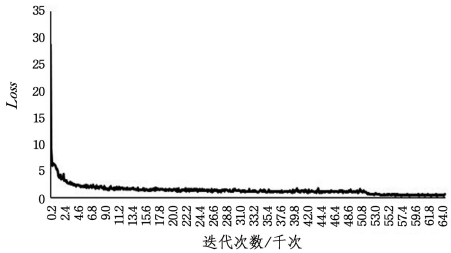

图 3 训练过程中的 loss 值变化

Fig. 3 loss value change during training

训练完成后得到的 YOLO 模型将用于在网页截图上识别页面元素。在本文实验过程中将置信度经验设置为 0.3，置信度大于 0.3 的页面元素将会被识别标记，并被转化为缺少功能属性的不完整的三元组〈类型，功能，位置〉，该三元组中的类型属性为上文提到的 32 种常见元素类型中的一种，位置属性包含元素识别框左上角的位置坐标信息和标识框的长宽信息，同时由于该阶段缺少页面元素周围的文本信息，因此此时的功能字段置为“空”。

4.4 实验结果与分析

4.4.1 不同版本的 Web 应用页面元素识别实验

针对问题 1，本节从云端软件开发协作平台 Gitee 中收集了 1 311 张样本图片，严格按照 3.1.1 节中的 Web 页面元素类型定义进行标注处理。本次实验标注了表 1 所列的 32 种元素及图标类型。

通过训练后，得到的模型在测试集中的元素识别精准度如表 7 所列。从表 7 中可以看出，不同类型的元素识别准确率差异较大，但整体上识别准确率较高。值得注意的是，刷新图标的识别精度较低。一方面，由于 YOLO 算法自身的局限性，导致其对小目标的漏检率较高，而 Web 页面中的刷新图标一般偏小，这使得此类元素识别效果较差；另一方面，训练

集与测试集中此类元素样本数量较少，也可能是导致这一结果的直接原因。为了进一步验证本文提出的 Web 页面元素识别方法能否在同一 Web 应用 GUI 发生变化的不同版本中准确识别出页面中元素的类型并定位其位置，本节选取 3 种 Web 应用 Mrbs，OrangeHRM，MantisBT 各自 GUI 差异较大的两个版本作为实验对象，使用本文方法来进行识别，以检查该方法识别页面元素的可行性。每个 Web 应用选取 5 张页面截图进行评价，实验结果如表 8 所列(所选实验截图内各元素数量如表 9 所列)。由表 8 可以看出，对比 3 种应用的两个 GUI 差异较大的不同版本可以发现，虽然 Web 应用页面的背景、元素大小和位置均发生了变化，但本文方法仍能正确地识别页面内的主要元素类型，但某些元素的识别效果欠佳，可能与数据集规模较小有关。

表 7 测试集中元素类型识别的平均精度

Table 7 Average precision of element type recognition in testing set

(单位：%)

元素类型	平均精度	元素类型	平均精度
下拉选择框	96.93	进度条	99.82
输入框	97.45	标签	100.00
按钮	97.43	通知	94.17
表格	98.98	消息	99.67
分页框	75.34	折叠	92.98
搜索框	83.73	添加	43.46
日期选择框	96.46	关闭	79.63
导航栏	98.51	设置	54.55
刷新图标	16.67	分类	100.00
图表	66.67	编辑	80.83
标签页	93.38	删除	78.71
多选框	91.92	上传	100.00
单选框	41.34	返回	100.00
日历	100.00	主页	100.00
弹框	100.00	退出	87.50
开关	92.18	工具栏	100.00

表 8 3 种 Web 应用两个不同版本部分元素的识别召回率

Table 8 Element recognition recall rate of two different versions of three Web applications

(单位：%)

Web 应用	按钮	输入框	单选框	下拉选择框
Mrbs-1.7.5	30.0	70.0	22.2	100.0
Mrbs-1.9.4	59.1	88.9	38.5	100.0
OrangeHRM-3.1.2	76.2	100.0	—	100.0
OrangeHRM-4.8	76.9	83.3	—	100.0
MantisBT-1.3.1	69.2	71.4	—	25.0
MantisBT-2.25.2	55.6	76.5	—	100.0

表 9 3 种 Web 应用两个不同版本部分元素的数量

Table 9 Number of partial elements in two different versions of three Web applications

Web 应用	按钮	输入框	单选框	下拉选择框
Mrbs-1.7.5	10	20	9	4
Mrbs-1.9.4	22	18	13	3
OrangeHRM-3.1.2	21	11	—	9
OrangeHRM-4.8	13	12	—	8
MantisBT-1.3.1	13	14	—	4
MantisBT-2.25.2	9	17	—	7

4.4.2 不同 Web 应用的页面元素识别实验

针对问题 2，我们选取了 4 个不同的 Web 应用作为实验对象，利用本文提出的页面元素识别方法对其进行识别，验证本文方法在不同 Web 应用中识别元素的可行性，每个 Web 应用选取 5 张页面截图进行评价，实验结果如表 10 所列，所选实验截图内各元素数量如表 11 所列。

表 10 不同 Web 应用部分的元素识别召回率

Table 10 Element recognition recall rate of different Web applications

(单位：%)

Web 应用	按钮	输入框	下拉选择框	导航栏
MyAdmin	86.8	81.8	50.0	100
VUE-Manage-System	100	100	42.9	100
GABS	84.6	71.4	0.0	100
常春藤物业缴费系统	50.0	92.9	100	100

表 11 不同 Web 应用部分元素的数量

Table 11 Number of elements in different Web applications

Web 应用	按钮	输入框	下拉选择框	导航栏
MyAdmin	53	11	8	4
VUE-Manage-System	11	7	7	4
GABS	13	7	1	3
常春藤物业缴费系统	12	14	1	4

其中，GABS 对下拉选择框的识别效果不佳，一方面是由于框中向下箭头不明显，本文方法将此元素误识别为按钮，另一方面是由于样本数量太少(截图内只存在一个下拉选择框)。总的来看，本次实验训练的 Web 应用页面元素识别模型能够较为准确地识别各类元素类型，且能够较为精确地识别出各类主要元素的边界，可在各类 Web 应用中有效地进行元素识别。

4.4.3 Web 应用页面元素图像文字融合实验

针对问题 3，本次实验利用百度的 OCR 接口对 Web 应用页面内的文字信息进行识别，并利用图像文字融合方法获取页面元素三元组〈类型，功能，位置〉。

本次实验使用百度的网络图片文字识别接口，该接口利用 PaddleOCR 深度学习开源框架提升检测效果，能够返回图片中的文字信息以及文字所在的位置，进一步提升了文字识别的准确率。结合 3.1.2 节得到的元素识别结果，利用 3.1.3 节中的页面元素功能信息识别方法，识别页面中各类元素的功能信息。图 4 给出了 Web 页面中的文字识别及页面元素功能信息识别的效果。

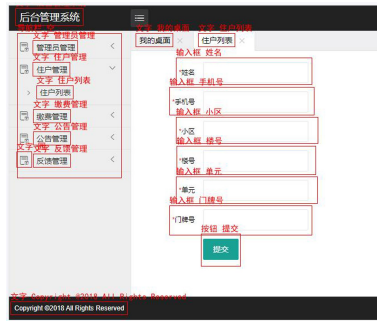

图 4 Web 页面中的文字识别及页面元素功能信息识别的效果

Fig. 4 Character recognition in Web page and effect of page element function information recognition

从图 4 可以看出，页面元素功能信息识别方法可以准确地识别出页面中的输入框、下拉选择框、日期选择框、表单和

按钮等功能信息，并能与页面元素位置及类型关联起来。因此，本文方法能够正确识别目标元素的类型和功能信息，并可以准确定位目标元素所在的位置，验证了本文基于机器视觉的 Web 页面元素识别方法的可行性。

4.4.4 基于测试需求描述的 Sikuli 可视化测试脚本生成实验

本节实验的实验对象如下。

测试网址：常春藤小区物业缴费系统

测试需求描述：

〈输入，输入框：姓名，“admin”〉

〈输入，输入框：密码，“admin”〉

〈点击，按钮：登录〉

〈点击，标签：住户管理〉

〈点击，标签：住户列表〉

〈点击，按钮：添加〉

〈输入，输入框：姓名，“test”〉

〈输入，输入框：手机号，“13166668888”〉

〈输入，输入框：小区，“Tichenpin”〉

〈输入，输入框：楼号，“6”〉

〈输入，输入框：单元，“2”〉

〈输入，输入框：门牌号，“302”〉

〈点击，按钮：提交〉

利用 3.2 节中的基于启发式搜索的可视化脚本生成方法在 Web 页面中搜索目标元素截图并进行替换，最终自动生成的 Sikuli 脚本，如图 5 所示。

图 5 自动生成的 Sikuli 测试脚本

Fig. 5 Automatically generated Sikuli test script

经过测试，该测试脚本能够正常按照给定的文本操作描述对 Web 应用进行各类点击和输入操作。实验结果表明，测试人员只需输入测试需求描述，本文方法就能够正确生成对应的可直接运行的测试脚本，极大地降低了测试脚本开发的门槛。

4.4.5 实验结果讨论

上述实验验证了本文基于机器视觉的 Web 页面元素识别方法的可行性，但同时我们也注意到多种因素影响着 Web 页面元素的识别效果，通过分析筛选得出以下影响识别效果的因素。

(1)类型的定义。Web 页面元素存在多种类型，元素类型的定义越准确，根据视觉属性的划分越细致，最终的识别效果就越好。

(2)数据集。数据集的数量以及质量影响着最终识别效果的好坏，数据集的数量越大、覆盖的范围越广，则训练出来的模型泛化能力越好。并且要求样本中包含大量的可识别元素，数据集中的样本质量越高，最终训练出的模型的识别效果就越好。搜集的数据集中会存在大量相似的样本，因此还需要对样本图片进行降重操作。

(3)数据集的标注规则。数据集中的标注效果也影响着模型的识别效果。在对数据集进行标注时应尽量框住元素的边框。如果对数据集样本中元素的标记不精准或与元素本身的大小相差太大，则均不能准确地识别元素的类型和大小。

(4)算法的选型。不同的目标检测算法对不同的对象有不同的识别效果，部分算法对大目标的识别效果较好，另一部分算法对小目标的识别效果较好。同时，由于不同的算法采用不同的模型，因此在训练和识别速度上也有所不同。对于 Two-stage 类算法会花费更多的时间进行训练，而 One-stage 类算法则会花费更少的时间，识别准确率也是可接受的。

(5)图标。为了增加 Web 页面的美观度，Web 页面中存在大量的图标类元素，每种元素都有其特定的功能，如返回、关闭、编辑、删除、下载、上传图标。在测试过程中这些图标元素也起着很大的作用，但由于此类图标一般较小且不同 Web 应用间外观差异较大，因此本文方法对此类目标的检测效果还有待增强。

5 相关工作

5.1 基于 DOM 的页面元素识别

目前应用广泛的自动化测试工具的原理主要是基于 DOM 树的操作，通过 DOM 对象识别元素并针对它完成测试工作，例如 Selenium 工具。Selenium 主要用于 Web 应用程序的测试，可以通过 CSS 选择器以及 XPath 定位目标元素，并可以直接在浏览器中像用户操作一样进行测试操作[19]，可以用在各种浏览器上，检测 Web 应用对该浏览器是否兼容，以及通过创建回归测试检测应用程序的功能。该工具被广泛运用于各种测试中，然而该工具的使用较为复杂，对测试人员的编程水平要求较高，存在编写的测试用例难以维护的问题。

QTP[20]是一种以关键字驱动为理念，用于功能测试和回归测试的自动化测试工具，简化了测试用例的创建和维护过程。它采用录制屏幕的方式产生脚本，在录制屏幕的过程中识别基于 DOM 树的控件对象，帮助测试人员做了许多工作，提高了测试人员编写测试用例的效率。

KylinTOP[21]是一种集性能测试、UI 自动化测试于一体的智能化驱动的测试工具，提供了快速生成测试用例、智能元素定位、步骤智能等待等一系列功能，很好地适应了软件敏捷开发的需要。通过使用该工具，使用者只需要关注业务操作即可生成测试用例，即使软件开发过程中产生快速迭代和功能变更，也依然能够执行测试，使得软件自动化测试更加智能。

WTF PageObject Utility Chrome Extension[22]自动化工具根据元素的ID、CSS和XPath定位器创建页面对象，把页面元素定位和元素操作方法按页面抽象出来，减少了代码的冗余，将业务和实现进行分离，降低了代码的维护成本。

5.2 基于模版匹配的页面元素识别

为了降低自动化测试编程的学习成本，应对软件快速迭代和变更的需求，提高软件开发过程中的测试效率，提高测试用例的可维护性和可读性，一些研究提出了基于模板匹配的测试用例生成方式。该方法通过图像对元素进行识别定位，关于此类方法的研究如下。

MIT的研究人员设计了一种新颖的图形脚本语言Sikuli，测试人员只需要具备基本的编程技能，利用截取的图片组合出测试脚本。该方法解决了基于DOM的元素控件查找可能无法找到的问题，它采用模板匹配的方法在屏幕中匹配截取的目标控件截图便可定位目标元素，大大降低了编写测试用例的难度，同时也提高了测试用例的可维护性。随着该方法的提出，Leotta[11]等提出了一种将Selenium网络驱动程序测试套件自动迁移到Sikuli测试套件的方法。Sikuli应用了OpenCV中归一化相关系数的模板匹配技术，将图案视为模板，并将模板与输入图像中等同大小的区域进行比较，以找到与模板最相似的区域，从而识别目标元素，但这种模版匹配方法比较简单，即使元素外观只发生了非常细微的变化，也会导致匹配的失败，使原有测试脚本无法正确执行。

修复损坏的Web元素定位器是Web测试用例的主要维护成本。为了检测可能的修复，测试人员通常会通过GUI检查测试与被测试应用程序的交互。现有的自动化测试修复技术专注于代码，而忽略了应用程序的可视化方面。Stocco等[2]提出了一种新的测试修复技术Vista，利用计算机视觉和本地爬虫来自动修复破损的Web测试脚本，但其实现方法中使用快速归一化交叉相关模板匹配算法来实现Web元素的检测，仍有许多现实情况无法处理，如分辨率变化等，存在一定的不足之处。

AirtestProject[23]是网易自己研发的解决游戏自动化测试的解决方案，是一个基于图像识别的跨平台自动化测试框架，受MIT研究院Sikuli成果的启发，是一种基于图像而不是控件对象的测试方式，支持在Android、IOS、Windows、小程序上进行测试，极大地提高了UI自动化测试的效率，但本质上同样使用基于模版匹配的元素识别方法，应用场景受限。而本文提出的基于计算机视觉的元素识别方法可有效适应Web应用UI层面的变化，泛化能力更强，适用于Web应用页面元素识别的问题。

5.3 测试脚本自动生成方法

测试脚本的编写是一项劳动密集型工作，每当程序发生变化，测试人员都需要编写大量的测试代码来验证代码的正确性，并且测试用例中包含大量重复的代码，该过程对于测试人员来说是一项重大的挑战[24]。因此，也有大量研究致力于测试脚本自动生成方法的研究。南京大学的Jiang等[25]提出了一种基于RFT的自动生成测试脚本的方法，提高了测试工作的效率。Liu等[26]也根据软件控件的属性和操作建立控件库和关键字库，通过测试文档进而生成测试脚本。本文借鉴了现有研究成果，把每一条测试需求三元组作为关键信息，使用启发式搜索方法生成可视化Sikuli脚本。

结束语 本文针对Web页面元素的多样性、差异性和可塑性等特点，提出了基于目标检测的Web页面元素识别方法，与现有的基于模板匹配的页面元素识别方法相比，本文提出的元素识别方法对复杂应用场景的适用性更强。在此基础上，针对测试人员给定的测试需求描述，提出了一种可视化脚本生成方法，通过对测试需求与页面元素三元组进行相似度匹配来获取页面元素截图，自动生成测试脚本，降低了测试脚本开发的门槛，减少了人工参与，进一步提高了测试效率。

在未来的工作中，我们计划继续扩大训练集，进一步提高页面元素识别的准确率，并识别更多类型的元素，还将优化自动化生成脚本的策略，提升自动生成脚本的成功率，进一步扩大本文方法的应用场景。

参 考 文 献

[1] CHANG T H, YEH T, MILLER R C. GUI Testing Using Computer Vision[C]// Proceedings of the 28th International Conference on Human Factors in Computing Systems. 2010: 1535-1544.

[2] STOCCO A, YANDRAPALLY R, MESBAH A. Vista: Web test repair using computer vision[C]// Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2018: 876-879.

[3] LOWE D G. Distinctive image features from scale-invariant keypoints[J]. International Journal of Computer Vision, 2004, 60(2): 91-110.

[4] REDMON J, DIVVALA S, GIRSHICK R, et al. You only look once: Unified, real-time object detection[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016: 779-788.

[5] Gitee[CP/OL]. [2021-12-10]. https://gitee.com/explore.

[6] GIRSHICK R, DONAHUE J, DARRELL T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation[C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014: 580-587.

[7] REDMON J, FARHADI A. Yolov3: An incremental improvement[J]. arXiv: 1804.02767, 2018.

[8] WANG H J, ZHOU B, PAN Y L. Ship Detection and Classification from High Resolution Optical Remote Sensing Images Based on YOLOv3 Neural Network[J]. Bulletin of Science and Technology, 2020, 36(3): 43-48, 58.

[9] KOVALENKO A, KOVALENKO A, SMIRNOV A, et al. DOM XSS Testing Technology Vulnerabilities[J]. Scientific and Practical Cyber Security Journal, 2018, 1(1): 38-45.

[10] ALTHOMALI I, KAPFHAMMER G M, MCMINN P. Automated visual classification of DOM-based presentation failure reports for responsive web pages[J/OL]. Software Testing, Veri-

fication and Reliability,2021,31(4). https://doi.org/10.1002/stvr.1756.

[11] LEOTTA M,STOCCO A,RICCA F,etal. Pesto:Automated migration of DOM-based Web tests towards the visual approach [J/OL]. Software Testing, Verification And Reliability, 2018, 28(4). https://doi.org/10.1002/stvr.1665.

[12] WANG J,TANG J,YANG M,et al. Improving OCR-Based Image Captioning by Incorporating Geometrical Relationship [C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021:1306-1315.

[13] Mrbs[CP/OL]. [2021-12-10]. https://github.com/meeting-room-booking-system/mrbs-code.

[14] OrangeHRM [CP/OL]. [2021-12-10]. https://github.com/orangehrm/orangehrm.

[15] MantisBT [CP/OL]. [2021-12-10]. https://github.com/mantisbt/mantisbt.

[16] MyAdmin [CP/OL]. [2021-12-10]. https://github.com/cdfan/my-admin.

[17] VUE-Manage-System [CP/OL]. [2021-12-10]. https://github.com/lin-xin/vue-manage-system.

[18] GABS [CP/OL]. [2021-12-10]. https://github.com/manier13579/GABS-AddressBookSystem.

[19] RAMYA P,SINDHURA V,SAGAR P V. Testing using selenium web driver[C]// 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT). IEEE,2017:1-7.

[20] XU T,ABBOTT R J,MILNE R I,et al. Phylogeography and allopatric divergence of cypress species (Cupressus L.) in the Qinghai-Tibetan Plateau and adjacent regions[J]. BMC Evolutionary Biology,2010,10(1):1-10.

[21] KylinTOP[CP/OL]. [2019-04-18]. http://www.70testing.com/cloud/help/index.html.

[22] WTF-PageObject-Utility-Chrome-Extension[CP/OL]. [2019-04-18]. https://github.com/wiredrive/wtframework/wiki/WTF-PageObject-Utility-Chrome-Extension.

[23] AirtestProject [CP/OL]. [2019-04-18]. https://github.com/AirtestProject.

[24] YU S,FANG C,LI T,et al. Automated Mobile App Test Script Intent Generation via Image and Code Understanding[J]. arXiv:2107.05165,2021.

[25] JANG Y,ZHAO J B. An Approach to Automatic Generation of Test Automation Scripts[J]. Computer Technology and Development,2007,17(7):4-7.

[26] LIU H G,ZHU L L,SONG L X,et al. A Method of Automatically Generating Test Script:CN104484269A[P]. 2015.

LI Zi-dong, born in 1997, postgraduate, is a member of China Computer Federation. His main research interests include software testing and so on.

WANG Wei-wei, born in 1990, Ph. D, lecturer, is a member of China Computer Federation. Her main research interests include software testing, Web application testing and search-based test case generation.

(责任编辑:喻藜)