UIED: a hybrid tool for GUI element detection

Conference Paper · November 2020

DOI: 10.1145/3368089.3417940

5 authors, including:

Zhenchang Xing
Nanyang Technological University
167 PUBLICATIONS **3,842** CITATIONS
SEE PROFILE

Jieshan Chen
Australian National University
7 PUBLICATIONS **286** CITATIONS
SEE PROFILE

Chunyang Chen
Monash University (Australia)
81 PUBLICATIONS **1,479** CITATIONS
SEE PROFILE

UIED: A Hybrid Tool for GUI Element Detection

Mulong Xie
Australian National University
Canberra, Australia
mulong.xie@anu.edu.au

Sidong Feng
Australian National University
Canberra, Australia
u6063820@anu.edu.au

Zhenchang Xing
Australian National University
Canberra, Australia
Zhenchang.Xing@anu.edu.au

Jieshan Chen
Australian National University
Canberra, Australia
Jieshan.Chen@anu.edu.au

Chunyang Chen
Monash University
Melbourne, Australia
Chunyang.Chen@monash.edu

ABSTRACT

Graphical User Interface (GUI) elements detection is critical for many GUI automation and GUI testing tasks. Acquiring the accurate positions and classes of GUI elements is also the very first step to conduct GUI reverse engineering or perform GUI testing. In this paper, we implement a **U**ser **I**terface **E**lement **D**etection (UIED), a toolkit designed to provide user with a simple and easy-to-use platform to achieve accurate GUI element detection. UIED integrates multiple detection methods including old-fashioned computer vision (CV) approaches and deep learning models to handle diverse and complicated GUI images. Besides, it equips with a novel customized GUI element detection methods to produce state-of-the-art detection results. Our tool enables the user to change and edit the detection result in an interactive dashboard. Finally, it exports the detected UI elements in the GUI image to design files that can be further edited in popular UI design tools such as Sketch and Photoshop. UIED is evaluated to be capable of accurate detection and useful for downstream works.

Tool URL: http://uied.online
Github Link: https://github.com/MulongXie/UIED

CCS CONCEPTS

• **Software and its engineering → Software development techniques**; • **Human-centered computing → Graphical user interfaces**.

KEYWORDS

Object Detection, User Interface, Deep Learning, Computer Vision

ACM Reference Format:
Mulong Xie, Sidong Feng, Zhenchang Xing, Jieshan Chen, and Chunyang Chen. 2020. UIED: A Hybrid Tool for GUI Element Detection. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '20), November 8–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3368089.3417940

ESEC/FSE '20, November 8–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7043-1/20/11…$15.00
https://doi.org/10.1145/3368089.3417940

1 INTRODUCTION

GUI offers a visualized way to display information and to interact with the software application through graphical UI elements, such as widgets, images and texts. The development of GUI is critical and laborious. It involves many repetitive and time-consuming tasks, such as GUI code implementation and GUI testing. Plenty of researches working around GUI automation [3, 17, 18, 31, 33] and testing [21, 30] aim to facilitate the development process and relieve pains of developers. The foundation of these tasks is to identify GUI elements. There are two practices used to recognize elements in GUI, instrumentation-based methods [2, 14, 19] and image-based methods. Instrumentation-based approaches are based on intrusive scripts and require accessibility of the back-end program [4, 12]. However, they cost plentiful effort to write scripts and hard to use when the back-end code is unavailable [11, 16]. On the contrary, image-based methods are more generic and less intrusive as it only requires GUI image to detect GUI elements [18, 21, 32]. But, to our best knowledge, there is no effective off-the-shelf GUI element detection tool that user can use without any extra work. Therefore, we developed an interactive web-based computer vision toolkit, User Interface Element Detection (UIED), which provides quick detection and easy management of GUI elements from GUI image.

UIED is a user-friendly web application where user can upload their own GUI images and receive accurate GUI element detection results. Detecting GUI elements from GUI image resembles object detection task in natural scene. The process involves detecting the presence and spatial location of certain target from natural background in a digital image or video and then classifies the detected object. Similarly, in our case of GUI element detection, the purpose is to identify and extract GUI widgets, images and text from the GUI image, which can either be a screenshot or design drawing. We implement 5 latest state-of-the-art methods in UIED, including 2 old-fashioned computer vision methods (Xianyu [32], REMAUI [18]) and 3 deep learning methods (Faster-RCNN [23], Yolo v3 [22], CenterNet [10]). However, GUI elements have large in-class variance and high cross-class similarity, while GUI designs are packed scene and close-by elements, and mix of heterogeneous objects [7]. These characteristics make it inadequate to apply the aforementioned methods straightforwardly to perform accurate

detection. Therefore, we design a novel GUI-specific element detection approach based on old-fashioned computer vision methods. The methods can be divided into two parts, non-text elements and text elements. First, for the non-text elements, we leverage and innovate a set of image processing algorithms (e.g. flood-fill [29], connected component labelling [24]) to extract them and then classify them using a ResNet50 classifier [13]. With the consideration of GUI distinct boundary, shape, texture and layout, we adopt a top-down coarse-to-fine detection strategy compared to bottom-up edge/contour aggregation strategy in existing methods [18, 32]. Second, for the text elements, we apply a state-of-the-art deep learning scene text model EAST [34]. By synergy of our novel old-fashioned processes and mature deep learning classifier, our method achieves the state-of-the performance in GUI element detection.

UIED provides an interactive dashboard that allows user to edit the result, such as dragging and dropping the element to change location, adjusting element's shape and size, removing elements, etc. The tool collects all detected elements as a set of UI kits in which they can be reused later. After a series of editing, UIED allows user to export the result including the edited GUI and the corresponding element information (e.g., position, size, class, etc.). The exported results can be further developed in various works, such as UI2CODE [17, 18, 33] applications that aim to automate GUI development by generating corresponding code from GUI image directly, and GUI testing [21, 30].

This paper makes the following contributions:

- We implement 5 existing detection approaches and our GUI-specific detection method to acquire elements from GUI.
- We develop an interactive web application UIED that allows user to manage GUI elements easily and produces reusable detection results for further development.
- An informative investigation among professionals proving the value of accurate GUI element detection approach.

2 APPLIED DETECTION METHODS

We applied both existing old-fashioned computer vision based methods and deep learning models retrained on mobile GUI images in UIED. Old-fashioned computer vision based methods process images pixel by pixel without using machine learning techniques. They are easy to deploy and adjust as no time-consuming training required. We apply 2 existing GUI detection methods, Xianyu [32] and REMAUI [18]. On the other hand, deep learning achieves remarkable success in object detection research areas and is able to predict results fast, we hence retrain 3 representative and state-of-the-art approaches, Faster-RCNN [23] (two-stages), Yolo v3 [22] (one-stage), CenterNet [10] (anchor free) and deploy them on UIED. Note that to perform GUI element detection, we retrain the deep learning models on a large mobile app screenshot dataset Rico [8].

REMAUI: It is a GUI reverse engineering work that converts mobile GUI image into code, leveraging the off-the-shelf image processing algorithms from OpenCV [27] library. For detecting non-text GUI elements, it adopts a bottom-up strategy where it first uses Canny edge [5] detection to acquire primitive shapes and regions of image content (e.g, edge, contour) and then aggregate them into objects progressively. It applies a simple optical character recognition (OCR) tool Tesseract [25] to detect GUI text.

Xianyu: This is another GUI reverse engineering work developed by Alibaba to synthesis code from GUI images. It adopts a similar idea as REMAUI. To improve the non-text element detection, it leverages the flood fill algorithm [29] to identify the connected regions and filters out the noise from the complex background, combined with recursive horizontal/vertical slicing to obtain the GUI elements Tesseract is also used to detect GUI text.

Faster-RCNN: It is a classic "two-stage" method which involves two steps: detection and classification. It first generates a set of region proposals in which likely to contain objects by a region proposal network (RPN). RPN uses a set of user-defined anchor boxes with different aspect ratios and computes an objectness score to determine whether the box contains an object. It regresses the anchor boxes to predict the object's bounding box. Then it uses a CNN-based image classifier to categorize the detected objects.

Yolo v3: Unlike Faster-RCNN, YOLO performs region regression and object classification at once. It determines the anchor box aspect ratio automatically through clustering ground truth in the training dataset. It generates a gridding feature map through CNN and produces a set of bounding boxes for each grid. YOLO then computes the objectness scores, regresses the box coordinates and classifies the object in the bounding box at the same time.

CenterNet: Both Yolo and Faster-RCNN depends on anchor boxes to detect targets, whose performance is affected by the ratio aspect of these anchor boxes. They are also ineffective on objects with various shapes that cannot fit into these boxes. To address these limitations, CenterNet uses an anchor-free technique. It is a one-stage detection model that predicts the position of the top-left and bottom-right corners and the centre of an object.

3 OUR HYBRID APPROACH

With consideration of the characteristics of the GUI image, we propose a novel GUI specific element detection approach. Our method divides the detection task into two part: non-text element detection and text detection. We leverage old-fashioned computer vision algorithms for non-text region extraction, and deep learning models to perform classifications and text detection. The synergy reduces the disturbance of text when detecting non-text elements, and achieves the state-of-the-art performance for GUI element detection. Figure 1 shows the process of our approach.

Non-Text GUI Element Detection Unlike deep learning models that utilize statistical regression to predict approximate bounding box, old-fashioned computer vision method can detect the position and shape of objects more accurately due to its pixel-level image processing. But the existing old-fashioned methods usually adopt a bottom-up strategy that aggregates the fine details (e.g., edge or contour) into objects. Such idea suffers noise of trivial image content and tends to over-segment GUI elements, especially when the GUI has a complex background. Therefore, we propose a top-down coarse-to-fine approach based on old-fashioned computer vision techniques for non-text GUI elements detection. The process involves 3 steps. First, the approach detects layout blocks through flood-filling algorithm [29] combined with the Sklansky's algorithm [26] to acquire the blocks' outer boundaries and produces a block map, as shown in Figure 1 (c) where different colour regions stand for potential different layout blocks. Then we use a

Figure 1: The overall process of our approach, where (a) is the input GUI, (b) is the text detection result by EAST, (c),(d),(e) is non-text elements detection and (f) is the merged final result.

shape recognition algorithm [20] to select rectangular regions and count them as GUI layout blocks. Second, the method generates a binary map by a simple but efficient binarization method based on the gradient map of the input GUI. If a pixel's gradient with neighbouring is small, it is regarded as a background point and coloured as black, otherwise white. Then, we segment the binary map into block segments based on previously detected blocks, as shown in Figure 1(d). In each block binary map, we detect GUI elements by connected component labelling algorithm [24] and use Sklansky's algorithm again to determine the elements' boundary. Third, we train a ResNet 50 [13] classifier on 90,000 GUI element instances with 15 categories to classify the extracted elements.

GUI Text Element Detection We determine GUI text as a scene text and apply the state-of-the-art deep learning scene test detector EAST [34] to detect text in the GUI image. It first feeds the input image into a feature pyramid network [15] and then computes six values for each point based on the final feature map to detect text (objectness score, top/left/bottom/right offsets and rotation angle).

4 WEB IMPLEMENTATION

UIED toolkit is a web application that provides the user with a convenient tool to detect and manage GUI elements in GUI images. It can export the detection results that can be further used in other applications such as GUI testing and GUI automation. In UIED, we integrate all of the previously mentioned approaches, including old-fashioned computer vision and deep learning methods. This tool also offers an interactive dashboard where the user can edit and manage the detection result. We implement Xianyu [32] and our own approach in OpenCV [27] and customize deep learning models in Tensorflow [1] and Pytorch [28]. There are two major parts of the UIED: the landing page and the dashboard.

Landing Page Figure 2(a) shows an illustration of our landing page, which displays the basic information and usage of UIED. Users are able to input a GUI image to be processed. They can either select example GUIs we provide to check the effect of detection and experience the basic usage of the dashboard or upload their own GUI to detect GUI elements. Furthermore, for our method, we allow the user to change some key parameters by slide bars to adjust the detection result. To facilitate image transmission in the server, we adopt a serializing structured data method, Google's Protocol Buffers method [9], which encodes the image into a buffer bytes.

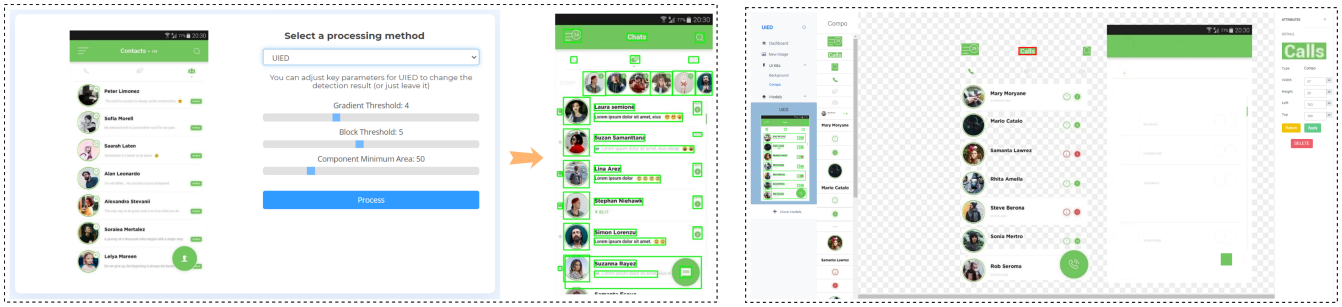
Dashboard UIED disassembles the input GUI into draggable GUI elements according to the detection result and displays them on the dashboard (in Figure 2(b)). In the dashboard, we implemented several functionalities to provide a more user-friendly interaction experience, including:

Drag & Drop: The user can adjust the position of GUI element to manually correct the detection result through dragging the element and dropping to somewhere else.

Attributes Management: When clicking a GUI element, the user can easily access the attributes of the element (e.g., type, width, height, left, top). We provide users with the ability to quickly edit the element, such as applying a new size and precise position, delete existing ones and withdrawing changes.

UI Element Kit: UI kit is the most efficient and profitable way to build a rapid GUI [6]. Therefore, we store all the detected elements in the dataset for the user to reuse. If the user further processes the input GUI in another method, the new GUI elements will also be added to the UI kits.

Detection Result Export After adjustment, the user can export the GUI element information in the compounded image, including position, size and class of elements. This information will be stored in a JSON file and is easy to use in further applications.



(a) Landing Page (b) Dashboard

Figure 2: Illustration of our UIED web application.

5 EVALUATION

The goal of our study is to evaluate the usefulness of UIED in terms of (i) its effectiveness in detecting elements and (ii) the usability of downstream tasks.

Effectiveness Measurement: Regarding the effectiveness of element detection in multiple approaches, we conduct experiments on 5k Android mobile GUIs collected from Rico [8]. This part is also published in our previous work [7] where we present a more detailed analysis. The main evaluation metrics we used is the F1-score which is interpreted as a weighted average of the precision and recall. Note that the measurements are evaluated on IoU > 0.9, where the IoU is the intersection area over union area of the detected bounding box and the ground-truth box. We further measure the cost of time to show the efficiency of each approach. Table 1 shows the performance of all approaches. The existing old-fashioned detection methods perform poorly (REMAUI F1=0.183, Xianyu F1=0.106) and the deep learning models gain better performance (Faster RCNN F1=0.271, YOLOv3 F1=0.249, CenterNet F1=0.282). Our approach achieves state-of-the-art performance (F1=0.524). Note that part of the reason why the score is not as high as expected is that the dataset itself is not perfectly precise. Combining the multiple models and the ability to manual adjustment, UIED is able to produce more accurate detection result as shown in Figure 2(a)

Usability Measurement: Regarding the user experience of UIED, we create a survey on 10 professional developers and researchers who come from GUI related work. They are asked to use UIED and questioned about the tool's usefulness for their work, as well as the future potential and extension of the tool.

Among those professionals, three of them are working on GUI reverse engineering research that synthesizes GUI code from GUI image (UI2Code). All of them indicated the significance of accurate GUI element detection for generating high-quality code while they have no off-the-shelf detection method to use. Similar situation in the other four participants who are researching robotic GUI automatic testing. They want to apply the robot arm to simulate human tester to test the mobile apps without writing any testing script, which means the visual information is critical. These researchers stated that although the GUI element detection producing exact element information to direct the robot tester is vital, there is no mature and accurate existing domain-specific method. And they believed an easy-to-use tool like UIED would be "more than helpful".

Table 1: Results of object detection (IoU > 0.9) and runtime efficiency

Approach	F1-score	Avg Time
YOLOv3	0.249	0.22s
Faster-RCNN	0.271	0.38s
CenterNet	0.282	0.34s
Xianyu	0.324	1.2s
REMAUI	0.357	5.3s
UIED	**0.524**	4.8s

We also surveyed two web developers. They agreed that a web application that recognizes the GUI elements in their design drawing and allows them to edit the image is "interesting and helpful". They were also very interested in the further potential of UIED which would support online UI2CODE function in future and think such a tool can be practical assistance in web development.

6 CONCLUSION AND FUTURE WORK

In this demo, we present UIED, a GUI element detection toolkit which supports two old-fashioned computer vision approaches and three commonly used deep learning approaches. Furthermore, based on the distinct characteristics of GUI, we implement a novel approach that combines best practices for non-text GUI element and GUI text detection. We embed our approach in the UIED with the option to adjust the key parameters to best adapt to the given GUI image UIED also provides users with an interactive and responsive dashboard with various useful functionalities to optimize the detection result, such as drag and drop, size and class editor. Finally, it can export the edited GUI image and corresponding GUI element information for further usage. UIED was evaluated in aspects of detection accuracy and tool usefulness. The evaluation suggests that UIED is a good starting point of software engineering in GUI tasks.

For future work, UIED has significant potential to be expanded with other applications. For instance, our ongoing UI2CODE project that aims to synthesis code from a given GUI image will be added to the tool once mature. With code generation, this tool will be dramatically helpful in GUI development in the way that the designer can pass their GUI design to our tool and get the usable code efficiently. Also, we are planning to utilize the UIED as the detection part for automatic GUI testing, which is expected to be added as an extension in UIED in the future.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Lingfeng Bao, Jing Li, Zhenchang Xing, Xinyu Wang, and Bo Zhou. 2015. scvRipper: video scraping tool for modeling developers' behavior using interaction data. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 2. IEEE, 673–676.

[3] Carlos Bernal-Cardenas, Nathan Cooper, Kevin Moran, Oscar Chaparro, Andrian Marcus, and Denys Poshyvanyk. 2020. Translating Video Recordings of Mobile App Usages into Replayable Scenarios. In *42nd International Conference on Software Engineering (ICSE '20)*. ACM, New York, NY.

[4] Karl Bridge and Michael Satran. 2018. *Windows Accessibility API overview.* Retrieved March 2, 2020 from https://docs.microsoft.com/en-us/windows/win32/winauto/windows-automation-api-portal

[5] J. Canny. 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8, 6 (Nov 1986), 679–698. https://doi.org/10.1109/TPAMI.1986.4767851

[6] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery DC: Design Search and Knowledge Discovery through Auto-created GUI Component Gallery. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–22.

[7] Jieshan Chen, Mulong Xie, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, and Guoqiang Li. 2020. Object Detection for Graphical User Interface: Old Fashioned or Deep Learning or a Combination? arXiv:2008.05132 [cs.CV]

[8] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 845–854.

[9] Google Developers. 2020. Protocol Buffers | Google Developers. https://developers.google.com/protocol-buffers

[10] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. 2019. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 6569–6578.

[11] Google. 2019. *UI Automator.* Retrieved March 2, 2020 from https://developer.android.com/training/testing/ui-automator

[12] Google. 2020. *Build more accessible apps.* Retrieved March 2, 2020 from https://developer.android.com/guide/topics/ui/accessibility

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Feng Lin, Chen Song, Xiaowei Xu, Lora Cavuoto, and Wenyao Xu. 2016. Sensing from the bottom: Smart insole enabled patient handling activity recognition through manifold learning. In *2016 IEEE First International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, 254–263.

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

[16] Microsoft. 2016. *Introducing Spy++.* Retrieved March 2, 2020 from https://docs.microsoft.com/en-us/visualstudio/debugger/introducing-spy-increment?view=vs-2019

[17] Kevin Moran, Boyang Li, Carlos Bernal-Cárdenas, Dan Jelf, and Denys Poshyvanyk. 2018. Automated reporting of GUI design violations for mobile apps. In *Proceedings of the 40th International Conference on Software Engineering*. 165–175.

[18] Tuan Anh Nguyen and Christoph Csallner. 2015. Reverse engineering mobile application user interfaces with remaui (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 248–259.

[19] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F Cohen. 2011. Pause-and-play: automatically linking screencast video tutorials with applications. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. 135–144.

[20] Dilip K. Prasad, Maylor K.H. Leung, Chai Quek, and Siu-Yeung Cho. 2012. A novel framework for making dominant point detection methods non-parametric. *Image and Vision Computing* 30, 11 (2012), 843 – 859. https://doi.org/10.1016/j.imavis.2012.06.010

[21] Ju Qian, Zhengyu Shang, Shuoyan Yan, Yan Wang, and Lin Chen. 2020. RoScript: A Visual Script Driven Truly Non-Intrusive Robotic Testing System for Touch Screen Applications. In *42nd International Conference on Software Engineering (ICSE '20)*. ACM, New York, NY.

[22] Joseph Redmon and Ali Farhadi. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[23] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.

[24] H. Samet and M. Tamminen. 1988. Efficient component labeling of images of arbitrary dimension represented by linear bintrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10, 4 (1988), 579–586. https://doi.org/10.1109/34.3918

[25] Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. IEEE, 629–633.

[26] Satoshi Suzuki and KeiichiA be. 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing* 30, 1 (1985), 32 – 46. https://doi.org/10.1016/0734-189X(85)90016-7

[27] OpenCV team. 2020. https://opencv.org/

[28] Pytorch Team. 2020. https://pytorch.org/

[29] Shane Torbert. 2016. *Applied computer science.* Springer.

[30] Thomas D White, Gordon Fraser, and Guy J Brown. 2019. Improving random GUI testing with image-based widget detection. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 307–317.

[31] Tom Yeh, Tsung-Hsiang Chang, and Robert C Miller. 2009. Sikuli: using GUI screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*. 183–192.

[32] Chen Yongxin, Zhang Tonghui, and Chen Jie. 2019. *UI2code: How to Fine-tune Background and Foreground Analysis.* Retrieved Feb 23, 2020 from https://laptrinhx.com/ui2code-how-to-fine-tune-background-and-foreground-analysis-2293652041/

[33] Dehai Zhao, Zhenchang Xing, Chunyang Chen, Xiwei Xu, Liming Zhu, Guoqiang Li, and Jinshui Wang. 2020. Seenomaly: Vision-Based Linting of GUI Animation Effects Against Design-Don't Guidelines. In *42nd International Conference on Software Engineering (ICSE '20)*. ACM, New York, NY, 12 pages. https://doi.org/10.1145/3377811.3380411

[34] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. 2017. EAST: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 5551–5560.