UICVD: A Computer Vision UI Dataset for Training RPA Agents

Madalina Dicu[1][a], Adrian Sterca[1][b], Camelia Chira[1][c] and Radu Orghidan[2][d]

[1]*Faculty of Mathematics and Computer Science, Babeş-Bolyai University, 1 M. Kogalniceanu, Cluj-Napoca, Romania*

[2]*Endava Romania, 51 A. V. Voevod, Cluj-Napoca, Romania*

Keywords: UICVD Dataset, Robotic Process Automation, User Interface Recognition, User Interface Understanding, Computer Vision.

Abstract: This paper introduces the UICVD Dataset, a novel resource fostering advancements in Robotic Process Automation (RPA) and Computer Vision. The paper focuses on recognizing UI (User Interface) components of a web application which is not as well known as recognizing real objects in images in the field of computer vision. This dataset derives from extensive screen captures within an enterprise application, offering a rare, in-depth look at real-world automation and interface scenarios. For RPA, the UICVD Dataset helps in training the machine model of an RPA agent for recognizing various UI components of the web application which is the target of the automation process. In Computer Vision, it serves as an invaluable tool for identifying and understanding user interface elements, ranging from basic icons to intricate structural details. Designed to support a wide spectrum of research and development initiatives, the UICVD Dataset is positioned as a critical asset for technology advancements in automation and user interface recognition. Its extensive, detailed content and ease of access make it a promising resource for enhancing existing applications and inspiring innovations in RPA and Computer Vision.

1 INTRODUCTION

Robotic Process Automation (RPA) is an innovative technology in the dynamic field of computer science that is changing how companies optimize and manage their processes. This section of the article introduces RPA as an engine for improving accuracy, operational efficiency, and resource management in the digital world. It further highlights the importance of UI object identification in RPA, describing in detail how this complex interface interaction process opens up the path to more efficient automation and platform connectivity. Central to this discussion is the exploration of a dataset that aids in the advancement of both RPA strategies and object detection mechanisms.

1.1 Robotic Process Automation

Robotic Process Automation (RPA) is revolutionizing business process management by automating routine tasks usually performed by humans (Institute for Robotic Process Automation, 2015), (Hofmann et al., 2020). RPA platforms such as UiPath[1], Automation Anywhere[2], Blue Prism[3], Microsoft Power Automate[4] enable the automation of business processes by interacting with software applications' User Interfaces (UI) components like buttons, text inputs, drop-down lists, and tables. They operate by allowing RPA developers to easily identify UI components of the target application and write code snippets that perform actions on these components, thus forming automated business processes that can be executed repeatedly with different inputs. These RPA platforms praise themselves because they require low or zero code and most of the programming is done graphically.

RPA refers to those tools that operate on the User Interface aiming to perform automation tasks using an "outside-in" approach. The information systems are kept unchanged, compared to the traditional workflow technology, which allows the improvement using an "inside-out" approach (Van-der Aalst et al., 2018).

[a] https://orcid.org/0009-0001-3877-527X
[b] https://orcid.org/0000-0002-5911-0269
[c] https://orcid.org/0000-0002-1949-1298
[d] https://orcid.org/0000-0002-1450-2680

[1]https://www.uipath.com/
[2]https://www.automationanywhere.com/
[3]https://www.blueprism.com/
[4]https://powerautomate.microsoft.com/en-us/

Dicu, M., Sterca, A., Chira, C. and Orghidan, R.
UICVD: A Computer Vision UI Dataset for Training RPA Agents.
DOI: 10.5220/0012632600003687
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 19th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2024)*, pages 414-421
ISBN: 978-989-758-696-5; ISSN: 2184-4895
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

RPA technology improves work efficiency and accuracy by reducing human errors and executing tasks more robustly. These software robots mimic human actions like mouse clicks and data entry, interacting with various software to ensure increased productivity and reduced costs. By being available 24/7, they offer a continuous operational advantage over human users. Advanced RPA tools integrate AI to enhance capabilities beyond mere UI interaction, contributing to standardization and business process enhancement (Chakraborti et al., 2020; Rajawat et al., 2021).

In the realm of process automation, commercial RPA tools also offer *process mining*[5] and *process discovery*[6]. However, the optimization of these processes often requires human expertise for analysis and decision-making. Academically, there's substantial interest in optimizing Business Process Management (BPM) by automatically extracting process steps and converting them into software robot sequences. Research focuses on understanding the anatomy of tasks from natural language process descriptions of the process that details the executed routines. The automatic identification of the type of performed activities (manual, human interaction, or automated) from text documents while employing supervised machine learning techniques was investigated in (Leopold et al., 2018). To identify the existing relationship between various activities of a process, the authors of (Han et al., 2020) used long short-term memory (LSTM) recurrent neural networks to learn from process description documents (PDDs). (Ito et al., 2020) proposes a new grammar for complex workflows with chaining machine-executable meaning representations for semantic parsing.

The current development of RPA tools makes use of AI advances in routine identification and automation. Still, the use of human expert skills is required to analyze how the routines are executed on the application's UI. Another research area addresses the actual automation of routines by examining the actions performed by human users when executing their tasks using software applications.

Tools like *Robidium* (Leno et al., 2020) and *SmartRPA* (Agostinelli et al., 2020) represent a new generation of automation tools. Robidium focuses on discovering automatable tasks from UI logs and generating execution scripts, operating differently from traditional record-and-play RPA tools. This is a Software as a Service (SaaS) tool that implements the robotic process mining pipeline proposed in (Leno et al., 2021).

Robidium uses UI log files that consist of data and events that are not related to a specific task identified beforehand. Its architecture comprises a preprocessing step on UI logs that allows the routine extraction and discovery of automatable routines and then these are compiled into a UiPath script.

SmartRPA (Agostinelli et al., 2020), on the other hand, records UI actions into a log file for routine identification and enables the generation of executable RPA scripts, offering customization for manual user inputs. The tool uses its action logger to record UI actions on the actions system, Microsoft Office applications, or web browser (e.g., Google Chrome, Mozilla Firefox) into a log file, used as input for routine identification. The tool allows the generation of a high-level flowchart diagram that can be studied by expert users for potential diagnosis operations and to generate executable RPA scripts based on the most frequent routine variant. Some input fields of the selected routine variant can be personalized before executing the related RPA scripts, supporting those steps that require manual user inputs.

Furthermore, tools like Ringer (Barman et al., 2016), a Chrome extension web replayer and Rousillon (Chasins et al., 2018), which uses Ringer for developing web automation scripts, show the potential for automating user actions in web environments. These technologies record and replay user actions, facilitating complex web automation and demonstrating the versatility and potential of RPA in enhancing business processes. The continual advancement of RPA tools, combined with AI and machine learning, signifies a promising future for business process automation, offering unprecedented levels of efficiency and accuracy.

1.2 Automatic Recognition of UI Elements

RPA platforms like UiPath or PowerAutomate recognize UI controls in two ways, depending on whether the target business application is a Windows desktop application (i.e., usually these RPA platforms operate only on Microsoft operating systems, i.e., Microsoft Windows family) or is a web application. If the target application is a Windows desktop one, the RPA platform uses directly the Windows APIs which return a hierarchical model for the UI form (very similar to the DOM (i.e., Document Object Model) of HTML documents)) to identify the components of the UI (i.e., buttons, labels, text input fields, etc.). If on the other hand, the target application is a web application, the RPA platform uses a browser plug-in (extension) to access the DOM structure of the web pages of the target application. This browser plug-in can extract from

[5]https://www.uipath.com/product/process-mining
[6]https://www.uipath.com/product/task-capture

the DOM structure the tags corresponding to simple UI controls like <input type="button" >, <button >, <input type="checkbox" >. For more complicated UI controls like icons, grids, tables, and lists, things are more complicated since these can be realized with a large and diverse number of HTML tags. One option is to recognize these high-level UI controls using computer vision techniques. In this paper, we deal only with the second type of target applications, that is web applications, not Windows desktop applications.

Object Detection is a fundamental task in computer vision, that aims to locate and identify objects in images or videos by outlining their position and boundaries and by categorizing them. Techniques in this domain are divided into two categories: one-stage methods, including YOLO (Redmon et al., 2016), SSD (Liu et al., 2016), and RetinaNet (Lin et al., 2017), which concentrate on rapid inference, and two-stage methods, such as Faster R-CNN (Ren et al., 2015), Mask R-CNN (He et al., 2017), and Cascade R-CNN (Cai and Vasconcelos, 2018), known for their focus on enhancing detection precision (Zou et al., 2023).

The rest of the paper is structured as follows. In Section 2 we present research work related to ours. Section 3 presents the details of the UI dataset proposed by this paper. We outline its structure, the number of samples, and also how the dataset was constructed. Following, Section 4 presents possible usage scenarios of the UICVD dataset. Section 5 presents the architecture of a prototype RPA agent that uses the UICVD dataset and the paper ends with conclusions in Section 6.

2 RELATED WORK

In recent years, the detection of components in User Interface (UI) design has garnered increasing interest from researchers, leading to rapid advancements and notable contributions in the field. A significant dataset in this domain is the Rico (Deka et al., 2017) dataset, which encompasses over 72,000 unique UI screens from more than 9,700 Android applications. It is a comprehensive combination of visual, textual, structural, and interactive design data. The development of the Rico dataset involved a blend of crowdsourcing and automation, ensuring a wide range of UI designs.

Another well-known dataset is VINS (Bunian et al., 2021), designed to enhance the process of locating mobile UI design examples. VINS stands out for its comprehensive collection of UI designs, totaling 4,800 images, including abstract wireframes and high-fidelity screens. It features detailed annotations with bounding boxes covering 11 distinct UI components, such as background images, menus, pop-up windows, input fields, and various buttons. A significant portion of these images are derived from the Rico dataset. Additionally, there is the Enrico dataset (Leiva et al., 2020), an extension of Rico that refines 10,000 UIs from Rico into 1,460 UIs across 20 design subjects, utilizing deep learning for classification.

It is noteworthy that these datasets primarily focus on mobile interfaces, leaving a gap in datasets addressing desktop applications. Regarding datasets with an emphasis on desktop interfaces, we identified the Website Screenshots dataset (Dwyer, 2022). This dataset, which is freely accessible and hosted on the Roboflow website, consists of 2,412 screenshot images sourced from over 1,000 different websites. The organization of this dataset is methodical: it includes 1,688 images in the training folder, 482 in the validation folder, and 242 in the testing folder. Each image is in JPG format with a uniform resolution of 1024 x 768 pixels, which is a common desktop screen resolution. Accompanying these images in each folder is a corresponding CSV file. These files contain detailed annotations for various objects within the images, providing a rich source of labeled data for analysis and model training. The dataset is annotated across eight distinct classes: button, text, image, link, title, field, iframe, and label. This variety of classes enables a comprehensive analysis of common web elements.

The predominance of mobile-focused datasets like Rico, VINS, and Enrico reveals a significant gap: the lack of datasets specifically tailored for desktop or enterprise web applications. This gap is crucial, as enterprise applications often have more complex interfaces and functionalities than mobile applications. The distinct features of enterprise applications - such as sophisticated navigation patterns, dense information displays, and complex interaction models - necessitate a specialized dataset for more effective UI design and analysis in this area. Introducing a new dataset focused on screenshots from enterprise applications would not only fill this void, but also allow for the development of more nuanced and effective tools for UI analysis and design in the broader context of desktop and enterprise environments. This would be a significant step forward in the field of UI component detection, addressing the unique challenges and requirements of enterprise application interfaces.

3 STRUCTURE OF THE DATASET

In the field of computer vision and robotic process automation (RPA), identifying User Interface (UI) elements is a crucial step in automating business processes. Recognizing a gap in existing datasets, which predominantly focus on mobile interfaces and are unsuitable for the automation of desktop-based, business web applications, we have embarked on creating a new dataset specifically tailored for desktop web applications. This initiative aims to facilitate the automation of business applications by providing a dataset dedicated to desktop images, addressing a significant gap in the current literature.

3.1 Specifications of the New Dataset

Our dataset comprises PNG images, primarily in two resolutions: 1920x940 and 1902x922. These specific resolutions stem from the native display settings of the application we used to create the dataset. The majority of images are at the resolution of 1920x940, aligning with the most commonly used screen resolution in our desktop environment. This diversity in resolution ensures that our dataset is representative of real-world scenarios, where variations in screen sizes and resolutions are common.

3.2 Data Source

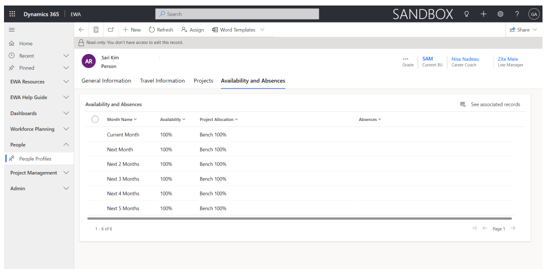
The dataset was created based on screenshots from an enterprise web application based on Microsoft Dynamics CRM and used within a multi-national software company. We captured a diverse range of screenshots from this application. Given the nature of enterprise applications, where structural variation between screens is minimal, many of our images may appear similar. However, this similarity reflects the real challenges in automating such applications. In total, we gathered 121 distinct screenshots.

3.3 Data Processing

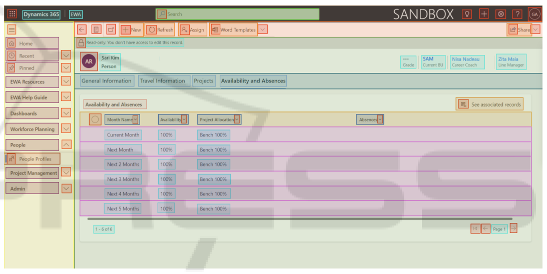
Our approach to data processing was minimalist, focusing mainly on anonymization to maintain confidentiality. No other significant processing techniques were applied to preserve the original structure of the application as much as possible. This approach ensures that our dataset accurately represents the typical environment of enterprise applications.

3.4 Data Annotation

The annotation and labeling of our dataset were carried out manually, using LabelStudio (Tkachenko et al., 2022), a versatile tool for annotating various types of data. Given the relatively small size of our dataset, manual annotation was deemed the most appropriate approach. Figure 1a displays a sample image from the dataset, while the second Figure 1b presents the same image, but in an annotated format.



(a) Image 24 from UICVD dataset.



(b) The annotated Image 24 from UICVD dataset.

Figure 1: Images from UICVD dataset.

Through extensive internal discussions and consultations with the company, we established 16 distinct classes to reflect both the current and future needs of the application. The dataset was then processed to include only significant elements for identifying UI components. Annotations are structured in a CSV file with the following information: image id, class, xmin, ymin, xmax, and ymax. In total, our dataset comprises 15123 annotations. Details of the classes and the number of annotations per class are presented in Table 1.

4 USING THE DATASET

The UI dataset can be used in at least the following two contexts: a) *training RPA agents in recognizing UI controls and understanding user interfaces* and b) *training machine learning models for recognizing UI components outside of the RPA context*.

Table 1: Distribution of classes and annotations in the UICVD dataset.

Class	Nr. of annotations
Icon	5704
TextLabel	4298
MenuItem	1138
Row	813
Button	757
SubmenuItem	685
NavigationItem	424
DropdownItem	379
InputField	149
SectionTitle	145
TitleBar	121
Menu	121
WorkingArea	121
VerticalMenu	110
NavigationMenu	100
TableHeader	58

4.1 Training RPA Agents in Recognizing UI Controls and Understanding User Interfaces

For an RPA agent that automates a business web application like a CRM (i.e., Customer Relationship Manager), ERP (i.e., Enterprise Resource Planning), or project planning application, it is very important to recognize various UI controls from the user interface of the application. UI controls like buttons, input text fields, drop-down lists, text areas, date controls, grids, tables, etc. are the entry points of the human user actions on the UI of the application. The human user operates on these UI controls by triggering various events like mouse clicks, text inputs, and mouse over, so an RPA agent must also operate on these UI controls to control the application.

For web applications, many such simple UI controls like buttons and input text fields can be easily recognized by an RPA agent that comes in the form of a browser extension. But some web applications customize these simple UI controls so much that they can not be easily identified from the DOM structure (i.e., some web applications want to create custom clickable buttons or checkbox/radio buttons and do not rely on the classical HTML controls <input type="button" >, <button >, <input type="checkbox" >). For these controls, it is very hard to recognize them by parsing the DOM of the HTML document and they are better identified by a computer vision approach. Also, complex UI controls like lists, tables, and grids are not easily recognized directly from the DOM structure of the HTML document - as these controls are usually realized by a variety of HTML tags together with a lot of CSS code. These complex UI controls are many times better recognized through a computer vision approach.

Our UI dataset is meant to be useful in training machine learning-based RPA agents for recognizing UI controls in the user interface of a business web application.

4.2 Training Machine Learning Models for Recognizing UI Components Outside of RPA Context

Outside the RPA domain, there is still the need to recognize icons and UI controls in a web application. Some possible scenarios where this would be useful are the following:

- screen reading for visually impaired people,
- advanced web scraping that does not rely only on XPaths in the DOM structure,
- automatic testing of a web application.

Our UI dataset could be used in training machine learning models that help visually impaired people use the target web application. It could also help describe the UI of the target web application for old people not familiar with digital technology.

Another possible usage scenario of our UI dataset is for advanced web scraping. Usually, web scraping programs navigate over to a specific URL and select a specific UI element from that HTML document based on a preset XPath in the DOM structure and they return the textual content of that UI element. More complicated web scraping scenarios can be supported if the scraping agent automatically recognizes the UI interface elements present in the HTML document loaded in the browser.

A very important and time-consuming phase of web application development is the testing of this application. Machine learning models previously trained on datasets of UI elements can automatically recognize UI components and could be used to perform automatic testing of the target web application.

5 PROTOTYPE RPA SYSTEM IMPLEMENTATION USING THE UI DATASET

The UICVD dataset is very useful for an RPA agent that automates a target web application. While such an RPA agent can easily recognize simple UI elements like clickable buttons and text input fields in a user interface through DOM analysis (if the RPA agent comes in the form of a browser extension), it will have a hard time recognizing clickable icons or complex, higher level, UI elements like grids and tables only by DOM analysis means. In this direction, a machine learning model can help the RPA agent recognize UI elements by computer vision means. We present in Fig. 2 the architecture of such an RPA agent that can use the UICVD dataset to apply computer vision techniques for recognizing and understanding user interfaces of web applications.

The RPA agent architecture outlined in Fig. 2 comes mainly in the form of a browser extension because the RPA agent targets/automates web applications (i.e., it needs to have access to the DOM of the HTML documents of the web application). The RPA agent also includes a back-end machine learning model that resides on a remote REST web server. The browser extension communicates with the back-end REST web server through XHR requests. The browser extension is always implemented in JavaScript and is made of the following type of code resources: *background scripts*, *content scripts*, and *popup script*.

The *popup script* is just the UI of the extension and it interacts directly with the human user. The *popup script* is an html+css+javascript bundle that presents a user interface to the user which assists the user in starting/ending the recording of an automated process, extracts for the user the semantics of the UI of the target web application (i.e., extracts and highlights the components (i.e., buttons, text labels, text input fields, grids, tables, icons, etc.) of the UI of the target web application), and allows the automated execution of a previously recorded process.

The *background scripts* is the JavaScript code of the browser extension that is loaded all the time across browser tabs and the *content scripts* is the JavaScript code that gets injected in the context of the document loaded in a browser tab. All these three components of the extension communicate with each other through message channels. The RPA agent has two main functionalities: a) discovering or recording a process that can be later automatically executed and b) executing automatically a pre-recorded process (i.e., automated flow). Both functionalities require code in both the *background scripts* and the *content scripts*.

First, a human user records a process with the assistance of the RPA agent (i.e., browser extension) and later the human user can automatically execute the pre-recorded process using the RPA agent; of course, the execution can be parameterized. This approach is also called programming-by-demonstration (Li et al., 2017). A recorded automated process is represented by a sequence of *browsing events* (usually mouse clicks required to navigate through the various menus of the target web application to reach the web page that supports a specific functionality of the target web application like adding a new user or updating a resource etc.) and also *business functionality events* like ‘inserting a text in a text field‘, ‘selecting a value from a drop-down list‘, ‘clicking a save button‘ etc. Of course, all *business functionality events* support parameterization (e.g. when executing the automated process, the user can specify what text value should be introduced in a specific text input field).

The *Executor* and *Process discovery/recording* components from the *background scripts* are responsible for coordinating the automatic execution and, respectively, the recording of a process. They are also responsible for injecting the *content scripts* in the document loaded in the browser. The components from the *content scripts* perform the direct work on the target web application. The *Executor* from the *content scripts* triggers click events on buttons and fills input fields in the browser so that the automated process is executed. The *Executor* component from *background scripts* takes each step from an automated process representation (i.e., automated flow), instructs the *Executor* component from the *content scripts* to perform the respective step (e.g. click event, text filling event), then it makes sure that the *content scripts* are loaded in the context of the current document (re-)loaded in the browser and moves to the next execution step in the automated process.

The *UI understanding* component in the *content scripts* is the most important component of the RPA agent and is responsible for understanding the UI of the target web application. It needs to detect simple controls like buttons, text input fields, and clickable icons, but also complex controls like grids so that it makes sense on the current operation/functionality that is facilitated by the UI of the web application. The RPA agent uses a dual approach for UI understanding: analyzing the DOM and performing a computer vision analysis on a picture of the document loaded in the browser.

To detect UI elements, the *CV analyzer* takes snapshot images of the document currently loaded in the browser and it sends these images to the REST web server through XHR requests. The REST web server runs a machine learning model on the snapshot image to detect UI elements like buttons, icons, input text fields, text areas, check and radio buttons, drop-down lists, tables, and grids and returns the detected UI elements as JSON objects to the *CV analyzer* of the browser extension. The *DOM analyzer* detects simple input controls directly from the DOM struc-

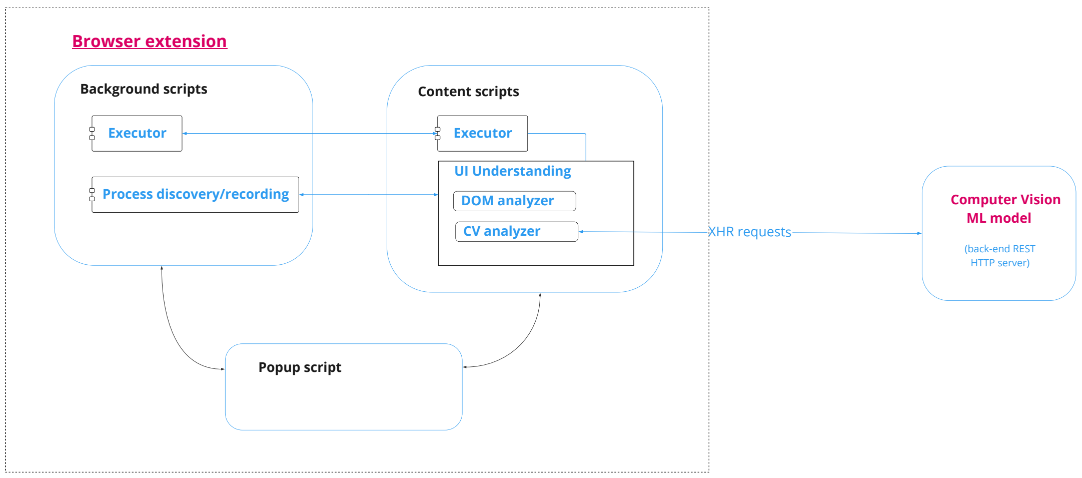


Figure 2: RPA agent architecture.

ture (e.g. buttons, text input fields, drop-down lists, links).

The machine learning model is developed in Python and is designed for object detection tasks, primarily utilizing the Faster R-CNN (Ren et al., 2015) framework, though it is compatible with other Convolutional Neural Network (CNN) architectures such as YOLOv8 (Jocher et al., 2023) for varied applications. The machine learning model was previously trained on the UI dataset.

The source code of the prototype RPA agent presented in this section is available for download from the following Git repository: https://github.com/KiralyCraft/WAPlugin/.

6 CONCLUSIONS

This paper highlights the UICVD Dataset as a valuable resource in the fields of Robotic Process Automation (RPA) and Computer Vision alike, demonstrating its wide applicability in both areas. It is noteworthy that, in the specialized literature, we have not been able to identify a similar dataset originating from an enterprise application. The process of data collection and the structure of the dataset have been elaborated, emphasizing two main applications, yet the possibilities for use are far more varied and interconnected.

The UICVD Dataset was created through the capture of screenshots from an enterprise application, providing a range of scenarios for both RPA and Computer Vision tasks. In the context of RPA, the UICVD Dataset supports building advanced RPA agents that recognize and understand the UI of enterprise web applications. In the field of Computer Vision, the dataset is distinguished by its ability to identify user interface (UI) elements - ranging from simple icons and buttons to more sophisticated structural components of the application, such as menus and data rows. Access to the dataset is provided via the URL below: https://github.com/MadaDicu/UICVD.

The paper also showcases a prototype implementation for a possible RPA agent that uses the UICVD dataset for UI understanding. We present its software architecture and detail its main components.

ACKNOWLEDGEMENTS

This research was funded by the ERDF through the Competitiveness Operational Programme 2014-2020, for the project ”Integrated System for Business Process Automation using Artificial Intelligence” (No. POC/163/1/3/121075), executed by ENDAVA Romania and Babeş-Bolyai University Cluj-Napoca.

REFERENCES

Agostinelli, S., Lupia, M., Marrella, A., and Mecella, M. (2020). *Automated Generation of Executable RPA Scripts from User Interface Logs*, pages 116–131.

Barman, S., Chasins, S., Bodik, R., and Gulwani, S. (2016). Ringer: web automation by demonstration. In *Proceedings of the 2016 ACM SIGPLAN international conference on object-oriented programming, systems, languages, and applications*, pages 748–764.

Bunian, S., Li, K., Jemmali, C., Harteveld, C., Fu, Y., and Seif El-Nasr, M. S. (2021). Vins: Visual search for mobile user interface design. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–14.

Cai, Z. and Vasconcelos, N. (2018). Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162.

Chakraborti, T., Isahagian, V., Khalaf, R., Khazaeni, Y., Muthusamy, V., Rizk, Y., and Unuvar, M. (2020). From robotic process automation to intelligent process automation: Emerging trends. *CoRR*, abs/2007.13257.

Chasins, S. E., Mueller, M., and Bodik, R. (2018). Rousillon: Scraping distributed hierarchical web data. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 963–975.

Deka, B., Huang, Z., Franzen, C., Hibschman, J., Afergan, D., Li, Y., Nichols, J., and Kumar, R. (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*, pages 845–854.

Dwyer, B. (2022). Website screenshots dataset. https://universe.roboflow.com/roboflow-gw7yv/website-screenshots.

Han, X., Hu, L., Dang, Y., Agarwal, S., Mei, L., Li, S., and Zhou, X. (2020). Automatic business process structure discovery using ordered neurons LSTM: A preliminary study. *CoRR*, abs/2001.01243.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Hofmann, P., Samp, C., and Urbach, N. (2020). Robotic process automation. *Electronic Markets*, 30(1):99–106.

Institute for Robotic Process Automation (2015). Introduction to robotic process automation. a primer. https://irpaai.com/wp-content/uploads/2015/05/Robotic-Process-Automation-June2015.pdf.

Ito, N., Suzuki, Y., and Aizawa, A. (2020). From natural language instructions to complex processes: Issues in chaining trigger action rules. *CoRR*, abs/2001.02462.

Jocher, G., Chaurasia, A., and Qiu, J. (2023). YOLO by Ultralytics. https://github.com/ultralytics/ultralytics. Accessed: December 1, 2023.

Leiva, L. A., Hota, A., and Oulasvirta, A. (2020). Enrico: A dataset for topic modeling of mobile ui designs. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 1–4.

Leno, V., Deviatykh, S., Polyvyanyy, A., Rosa, M. L., Dumas, M., and Maggi, F. M. (2020). Robidium: Automated synthesis of robotic process automation scripts from UI logs. In *Proceedings of the Best Dissertation Award, Doctoral Consortium, and Demonstration & Resources Track at BPM 2020 co-located with the 18th International Conference on Business Process Management (BPM 2020), Sevilla, Spain, Sept. 13-18, 2020*, volume 2673, pages 102–106. CEUR-WS.org.

Leno, V., Polyvyanyy, A., Dumas, M., Rosa, M. L., and Maggi, F. M. (2021). Robotic process mining: Vision and challenges. *Business & Information Systems Engineering: The International Journal of WIRTSCHAFTSINFORMATIK*, 63(3):301–314.

Leopold, H., van der Aa, H., and Reijers, H. (2018). In *Identifying Candidate Tasks for Robotic Process Automation in Textual Process Descriptions*, pages 67–81.

Li, T. J.-J., Azaria, A., and Myers, B. A. (2017). Sugilite: Creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 6038–6049, New York, NY, USA. Association for Computing Machinery.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer.

Rajawat, A. S., Rawat, R., Barhanpurkar, K., Shaw, R. N., and Ghosh, A. (2021). Chapter one - robotic process automation with increasing productivity and improving product quality using artificial intelligence and machine learning. pages 1–13.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Tkachenko, M., Malyuk, M., Holmanyuk, A., and Liubimov, N. (2020-2022). Label Studio: Data labeling software. Open source software available from https://github.com/heartexlabs/label-studio.

Van-der Aalst, W. M. P., Bichler, M., and Heinzl, A. (2018). Robotic process automation. *Business and Information Systems Engineering*, 60:269–272.

Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE.*