Computer Vision in UiPath's Automation Ecosystem: Automation Beyond Selectors

Himadeep Movva

Independent Researcher
movvahimadeep@gmail.com

Abstract:
Computer Vision in UiPath is a novel solution that lets the creation of automation bots in applications where the traditional selectors aren't available and rather rendered as image of the screen. This research paper discusses how Computer Vision helps resolve challenges related to image based automation. Also, it mentions how automation used to be configured in Citrix environments or VDI environments prior to introduction of Computer vision feature in UiPath, and the problems related to image based automation. Different features and techniques available within Computer Vision activity package including the available activities and how they can be leveraged while configuring automation is mentioned. As these activities are embedded in the UI automation package in UiPath, it is seamless to use and configure the CV automation.

Keywords: Computer Vision, OCR, Neural networks, DOM, UI Automation, GUI, VDI, Citrix, AI, and fuzzy logic.

1. INTRODUCTION:

UiPath Computer Vision activities are based on Machine Learning algorithms and can identify the elements on the screen and interact with them just like a human would do. In many desktop applications and web applications, screens follow DOM (Document Object Model) structure, representing the underlying GUI (Graphical User Interface). DOM properties allow UiPath to access selectors like html/body/div/button and identify fields, buttons, or other elements (Ex: id, name, type, class). DOM properties are usually available for web applications as HTML/CSS rendered by a browser , and Windows apps with WPF, WinForms, or .NET and expose UI Automation or Active Accessibility. DOM Properties will not be Available for Legacy Desktop Apps that have no UI built with modern accessibility layers, Citrix or VDI without Remote Run time as interface will be streamed as an image, RDP sessions without extensions, scanned documents or images with purely visual content and no structure and some other applications not mentioned here. When DOM is available, UiPath Bots can be built reliably and efficiently. However, if DOM isn't available, it is imperative to rely on UiPath's Computer Vision activities. A quick test to check whether DOM exists for a window is to click on indicate on screen in UiPath and point to the target application. If some tags of the selectors appear in the format "<html><body><div id='login'>", it means DOM exists. If only coordinates or screen regions are seen, it means that the DOM isn't available. Using a combination of techniques including deep learning for object detection, OCR for text recognition, fuzzy matching, and image recognition, UiPath's CV engine "sees" the interface and understands all the controls and text on it [1]. Using Computer vision, many applications that needed image classification to further drive automation can be automated [2].

2. EARLY AUTOMATION TECHNIQUES IN VIRTUAL ENVIRONMENTS:

Before native Citrix integration or Computer Vision, a remote app in Citrix/RDP screen was essentially an image rendered application. UiPath bots had to visually identify, use image-based activities, and navigate by looking for images or text and sending keystrokes. UiPath Studio's older version included a Citrix Recording wizard and various image/OCR-based activities to facilitate these actions. Image-based automation uses screenshot snippets of UI elements as references. During development, the developer indicates a UI element (button, icon, etc.) on the remote screen and the bot identifies this as an image template. During runtime, the bot searches the live screen for that image configured and, if identified, performs an action such as click or

hover. In the backend, this identification is based on the resolution with which the screen would be loaded, affecting the identification due to low bandwidth and other related constraints. UiPath provided activities, such as Click Image, Find Image, and Image Exists were used for these purposes. For example, a Find Image activity could halt execution until a specific image appears on screen (or timeout) before proceeding. Also , the evidence from research suggests that Object detection of UI elements using Machine Learning is viable and a better solution [3].
When using these activities, developers drag and select box around the target element in the Citrix/RDP window. Then, Bot stores that image and, at runtime, scans the screen's pixels for a possible match. If a match is found, the coordinates are returned. Following action, such as a click, can be offset relative to that image's location. The image match can be sensitive to slight visual changes and is prone to failure. As it relies solely on image recognition, identifying elements isn't a robust mechanism. It works only if the screen looks exactly as it was configured during development– any change in resolution, color, theme, or slight UI change could cause the image matching algorithm to fail. It is susceptible to even minor changes - a highlighted icon or background color change can break the automation. Image-based recognition is also prone to misidentifying elements if similar elements are visually on screen, potentially risking clicking on the wrong element. Because of these issues mentioned and several other challenges that entail image-based automation, this method is unreliable and is generally opted for only as a last resort. It also requires a lot of maintenance work. Suppose if the UI is updated, or the visual elements of an existing UI are updated without making significant changes to the underlying functionality, all reference images may need to be recaptured.
OCR-Based Text Automation: Text recognition via OCR (Optical Character Recognition) is another technique used in virtual sessions. OCR activities attempt to "read" text from the pixels on screen, enabling the robot to locate or extract information that isn't programmatically accessible. UiPath included activities like Get OCR Text, Find OCR Text, and Click Text/Click OCR Text, which use an OCR engine to identify given text in the remote interface. For example, a workflow might search for a specific label or button text on the Citrix screen and then click it by using OCR to find its coordinates. The OCR Text activity would look up and find if any given text is found within the application. The text input must match the text that needs to be found using OCR. UiPath leveraged OCR engines such as Google Tesseract, Omni Page, Microsoft, etc, to scan a specific indicated region and find text being searched for. During development, the developer provides a verbiage for an element such as a number, date, policy ID, etc. At run time, the OCR engine returns the exact position of the text, letting the bot click on that element. Usually, OCR is configured along with an anchor, as scanning the whole region would be time-consuming and complex instead of scanning for a specific phrase. Click OCR text is an activity that finds and clicks on text, eliminating the need for an anchor activity. The activity Get OCR text extracts text from the screen into variables. Achor base activity cannot be used as it is designed for selector-based automation. The anchor and target need to be identified as separate UI elements, so this approach may not work in VDI environments.
Keyboard and Hotkey Automation: Hotkey-based automation is often used in Citrix/VDI environments to offset the unreliability of screen scraping. This involves sending keystrokes, and using special keys to navigate through a remote application without using the cursor. Send hot keys can be used to get to a certain position and get information. Still, the challenge is that a slight change in the screen size or any movement of the cursor while performing operations on the screen could tamper with the expected places where the cursor needs to land. Hence, for this reason, using hot keys functionality as an alternative is not suggested and is highly prone to errors, as there's no way to verify the operation. Keystrokes can be sent in two ways: hardware keystrokes using the operating system or simulated keystrokes. If the remote window is not active, the simulate option will not work, necessitating that the active window exists for the simulate option. As much as possible, instead of clicking on icons, if the same action can be achieved using keyboard shortcuts, it is recommended. Although keyboard-based shortcuts eliminate the need to use image-based activities, there are challenges associated with this implementation. They include no reliable verification about where the send hotkey command is sent but the screen can be verified post the send hot key.Also, if the remote application's User Interface isn't stable, just sending keystrokes without verification could lead to wrong clicks, and eventually, incorrect steps in the process by the robot.
Synchronization and Error Handling in Citrix or VDI environments:
In Citrix or VDI environments, UI elements aren't accessible through selectors, and hence, UiPath developers relied on activities involving image recognition, OCR, and keyboard shortcuts. But these methods are less

reliable and heavily resource-intensive, necessitating a robust error-handling mechanism. There would be cases where bots try to interact with the element before it's fully loaded or visible. For example, submit button may not appear immediately due to network lag in Citrix, or the OCR (Optical Character Recognition) engine may take a variable amount of time to complete an action. To account for these situations, developers use retry scope, delays, image exists, loops with retry and refresh, and try-catch mechanisms. UiPath's early Citrix recorder included default mechanisms such as wait for specific image to appear before click, or capturing coordinates during development. But, these features are not adaptive and often static, forcing the developers to supplement the logic with an error handling mechanism. The changes in screen resolution, font, or color may break the automation, and, OCR is computationally expensive.
All in all, automation in Citrix or VDIs prior to Computer Vision and Native Citrix integration was fragile and prone to more errors. UiPath identified the pain points related to these aspects and addressed these challenges using advanced features of Computer Vision and Native Citrix. UiPath AI Computer Vision is a machine learning capability that allows robots to visually recognize and interface with UI elements on a screen like a human would. Computer Vision relies on the elements' visual appearance and context in the environments rendered as images without selectors.

3. HOW COMPUTER VISION ENABLES AUTOMATION BEYOND SELECTORS:

UiPath traditionally relied on selectors, which are structured strings that help identify User Interface elements. These selectors are retrieved from web applications' Document Object Model (DOM) tree and the Windows UI Automation Framework. Selectors contain attributes such as aaname (visible label), id, name, title, class, idx and more. However, identification using selectors fails in the case of Citrix/ RDP , which sends the UI as an image to the bot, providing no access to the underlying metadata related to elements. Also, selectors will either be unavailable or unstable in case of Legacy mainframe or custom thick-client applications that are built using nonstandard UI frameworks. UiPath Computer Vision is a set of AI-powered activities that allow robots to see the screens just like how a human sees the screen, understand elements based on appearance and layout, and interact with User Interface elements, even without reliable selectors or without selectors. UiPath Computer Vision uses a cloud-based or on-premises AI model with a vast amount of training data, probably in the size of a few million screens and their respective elements. The execution flow is as follows: a screenshot of the screen is taken, the AI model analyzes visual elements, the robot identifies elements based on spatial positioning relative to text, and the action is performed at the coordinates with high precision. Using visual anchors, robots detect static labels as anchors and search for associated elements relative to those anchors. In addition, dynamic region mapping allows bots to highlight UI controls based on learned visual patterns and not code structure, allowing CV to adapt to UI and layout changes. Virtual Desktop Infrastructure (VDI) platforms like Citrix, VMware Horizon, and Azure Virtual Desktop are commonly used to provide remote access to enterprise applications. Computer Vision technology has been specifically developed to overcome selectors' lack of availability in VDI environments. [4] Rather than relying on selectors, AI computer Vision uses AI including object detection, OCR, fuzzy matching, and image-matching techniques for recognition. Firstly, AI computer Vision performs element detection on the Machine learning server and OCR text detection, combining these methods into a full understanding of the UI. The relationship between elements detected with these two methods is then encoded into a multi-anchor descriptor, uniquely identifying the targeted element. The components are AI computer Vision activities that are now part of the UI automation package, server, on-prem or cloud-based, hosting an AI model that performs actual UI automation analysis. Although an on-prem server can be used, it is recommended that a cloud-based server be used for all AI computer vision and UI automation activities. Depending on the need and requirement, hosting and managing own on-prem AI computer Vision server can be used. By going to on-prem set up, firm needs to have its own hardware needs to be in place. Also, you need to deploy, update, and maintain your own environment locally. Compared to the UiPath cloud server, you might also encounter backward compatibility issues when upgrading the AI model. While there are many benefits of computer vision, one notable benefit is that it offers feature of run time auto scroll-support, letting the bot to easily automate scrollable content in webpages or apps.
Once the user installs the UI Automation package, Computer Vision Recorder wizard becomes available in the ribbon. The recorder isn't supported for cross-platform project types. This recorder helps in automatically generating workflows using computer vision activities. By clicking record button, recording starts and enables

the developer to select the application to automate, which will be used as a target for all computer vision activities. Once an application is indicated, the selected window is sent to the Computer Vision server, where it is processed, and the wizard enters recording mode. All other screen content besides the selected window is grayed out, and the recorder interface changes. It is shown in Figure 1. Type button lets the user to select text field to identify and type text and/or hotkeys into it. Using Click relative, an anchor can be configured. While the Get button extracts text from a certain area, the save and exit button lets the user close the recording session and generate the activities required to perform actions recorded in the designer panel. The stop recording button stops the recording and saves the recorded steps before clicking stop button.

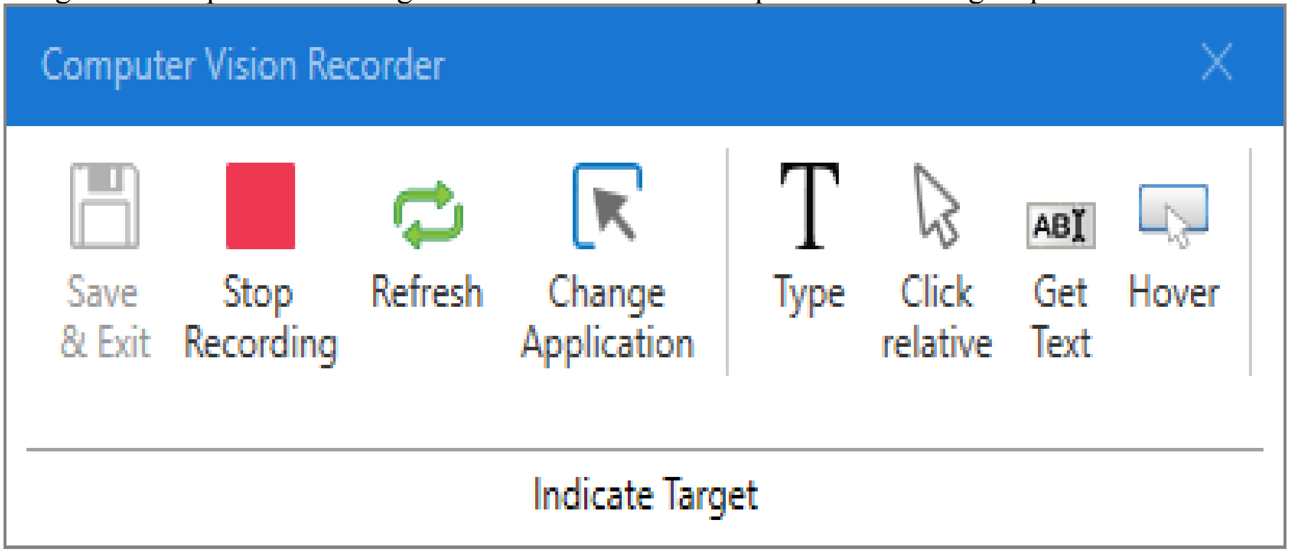

Figure 1- Computer Vision Recorder Wizard

It is important to note that Computer Vision activities are incompatible with Windows 7 version. [5] The Computer Vision activities contain refactored fundamental UI Automation activities such as Click, Type Into, or Get Text. The difference between computer vision activities and their classic counterparts is that CV activities use neural networks to identify UI elements such as buttons, text-type input fields, or checkboxes without using selectors. These activities bypass the issue of non-existent or unreliable selectors as they send images of the window user automates to the neural network, where it is analyzed, identifying and labeling the UI elements. All the Computer vision activities function under the CV screen scope activity, which establishes a connection to the neural network server and eventually helps analyze the UI elements for automation. As a first step, drag and drop CV Screen scope activity to the designer panel. Once this is done, select the area of the screen that you want to work in using the on-screen button in the body of the scope activity. Double-clicking the informative screenshot displays the image that has been captured and highlights in purple all the UI elements that the neural network and OCR engine have identified. Instead of selecting whole screen, area selection can also be limited to a portion of UI. This is helpful in cases where there are multiple text fields with same and hence there'd be difficulty in properly identifying the right field. On a CV screen scope is configured, all other activities can be used to build automation. Indicate field specifies what is being indicated at the current instance. When the helper wizard is active, the target needs to be shown. Upon selecting the target, the wizard automatically captures the anchor if it's available. The show elements button in the Figure 2- CV Get Text highlights all the UI elements that Computer Vision has identified. Screen refresh is helpful at design time when there are some changes in the target app, so that a new picture needs to be sent to the computer vision server for analysis. The computer Vision activities also offer support for indicating tables. To start with, target by selecting a cell, prompting neural networks to automatically identify the column and the row that define position of the cell. After successfully indicating the Target, the wizard closes and the activity is configured with the target you selected.

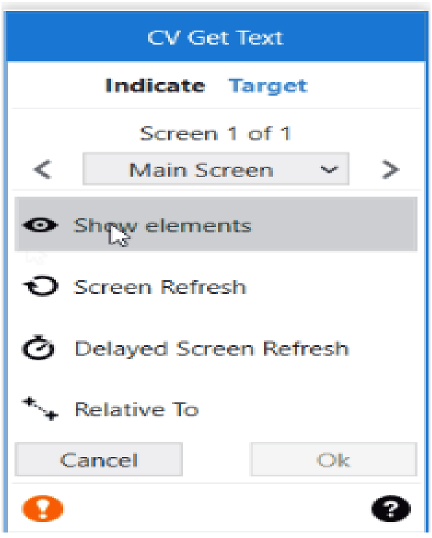

Figure 2- CV Get Text

The following are the available CV activities and their uses.

i. CV Check: It verifies UI elements using a CV descriptor. In short, it checks if a UI element exists based on visual attributes and returns a Boolean result indicating whether the element exists on the screen. It helps configure conditional logic in which, if a button appears, click it. CV Descriptor is a unique reference to a UI element captured during design time, with visual information such as relative position, text label, and element type. The output, as mentioned, is a Boolean variable, indicating whether the UI element is visually detected at run time. If this activity is included in Try Catch and the value of the ContinueOnError property is True, no error is caught when the project is executed. CV descriptor contains target, which is the main UI element, numbers representing a rectangle on the screen, and an offset point defining the precision point within the target bounding box. Also, CV Descriptor has one or more anchors with coordinates, helping identify target elements in case of multiple similar-looking elements.

ii. CV Click: It clicks a specified UI element targeted by the UiPath Computer Vision neural network. This activity is helpful when working with scrollable UI elements such as long lists, drop-downs, and tables. Sometimes, even if it's not visible initially, clicking on an element may be necessary. In CV, the elements can be identified by enabling the scroll option.

iii. CV Dropdown select: This activity allows you to select an item from a drop-down menu or list. The activity will interact with the currently active window if the Activate check box is not selected.

iv. CV Element Exists: This is based on the screen coordinates of the Target and each Anchor used, in case if it is used. These values will be stored in the CV Descriptor object. Once the Indicate On Screen feature is used at runtime, the CV Descriptor is automatically generated in this field and has the following structure:

```
"Target: Image (14,61,105,54) OffsetPoint: (-10,-75)" +
"Anchor: Text 'Anchor1' (41,36,19,9)" +
"Anchor: Text 'Anchor2' (75,36,37,9)" +
```

A reusable region refers to specific are on the screen, represented by rectangle, that can be shared across multiple Computer Vision activities to target UI elements within the same region. This is very helpful, especially when performing multiple actions on the same screen. Also, this helps avoid frequent use of indicate on screen using CV screen scope activity and improves consistency in the code reusability. Input Region accepts a rectangle variable that defines a pre-captured screen and will be used as the target area, ignoring any target or anchor set manually in the activity. Input region, once set, also overrides the CV descriptor. Suppose a developer identified a dropdown or button using CV Click or CV Text and saved its region in a variable element Region. In that case, it can be checked if it's still visible by using CV Element Exists with Descriptor. So, once set Input Region is equal to the element Region. Output Region outputs the target region where the element was found as a Rectangle variable. This can be passed into another CV activity (e.g., CV Click, CV Type Into, etc.) as an Input Region. We can use CV Element Exists with Descriptor to detect a UI element, and if found, click on it. Also, this avoids re-scanning the entire screen.

v. CV Extract Data Table: This activity extracts a table that is visible on the screen and stores it in a data table. In this activity the options, just as they are available in other CV activities, scrollable table and, number of scrolls, and scroll direction are very helpful. The number of scrolls decides how many scrolls need to be made while searching for a target element. If the table is scrollable, using scrollable table feature, bot scrolls down automatically and detects the end of the table. Also, scroll directions can be set among Up , Down, or None. Depending on the selection, the scroll will be performed at run time.

vi. CV Get Text: This activity extracts the text from a specified UI element. There is an option named Method to perform this activity. OCR type of method uses ocr engine that is specified in the CV screen scope of parent activity to retrieve the text. Select all type of method copies the entire text by using a clipboard. Select row method copies text in the entire row by using the clipboard.

vii. CV Highlight: Visually highlights a specified UI element. The element is identified by using the UiPath Computer Vision neural network. It's a visual debugging or validation activity that doesn't click, type, or extract data. It works based on a configured CV Descriptor and if nothing is found, nothing will be highlighted.

viii. CV Hover: This is used to trigger UI reactions that occur on hovering mouse such as revealing dropdowns or displaying hidden buttons. Option fine tuning of hover position can be achieved using OffsetX/OffsetY. Once the element is found, the mouse is moved to that element's center position.

ix. CV Refresh: It is used to re-capture and update the visual model of the UI window inside CV screen scope activity. This is typically used when the screen content has changed, and the bot needs to get the updated visual interface. As in VDIS or Citrix, UiPath doesn't have access to real time Document Object Model changes, CV refresh is needed. Using this activity, when new pop up appears, when dropdown expands, when a dialog box loads dynamically, or when screen updates after navigation, the update can be captured.

x. CV Screen Scope: This activity initializes UiPath Computer Vision neural network, analyzes the indicated window and provides scope for subsequent CV activities. We can choose which OCR engine to use with this activity. By default, UiPath screen OCR will be used. All CV activities like CV Click, CV Type Into, CV Get Text, CV Element Exists, etc., must be placed inside a CV Screen Scope in order to work. Without CV screen scope, bot cannot determine the target region for CV processing.

xi. CV Type Into: This allows robots to enter text into UI element based on visual recognition. The options in this activity include Target, which visually represents the field to type into, click before typing, which clicks into the field before typing, empty field, which clears existing content before typing, Input mode, secure text, and OffsetX/OffsetY.

4. CONCLUSION:

All in all, UiPath Computer Vision provides automation capabilities in Citrix or VDI applications where traditional selectors aren't available. It leverages the techniques related to anchor-based targeting by understanding contextual relationships, adapting to screen changes, and matching based on fuzzy logic that recognizes elements even when their visual appearance changes slightly. A unique advantage of Computer Vision activities is that there's option for dynamic scroll during run time to locate elements that are not visible on the screen. As CV relies on what is displayed on the screen, it is agnostic to operating systems, platform, or framework of the application. Hence, Computer Vision enables intelligent, reliable, and platform agnostic automation. By eliminating dependency on selectors for VDI or Citrix applications, it extends automation to new frontiers. This introduction of AI-driven vision into UiPath's ecosystem greatly enhances the flexibility of automation solutions, allowing companies to automate processes in healthcare, finance, and other industries even when facing older or remote applications that defy traditional automation methods

REFERENCES:

[1] [Online]. Available: https://docs.uipath.com/ai-computer-vision/automation-suite/2023.4/user-guide/introduction. [Accessed May 2025].
[2] S. H. R. M. &. J. D. K. Madakam, "The future digital work force: robotic process automation (RPA)," JISTEM-Journal of Information Systems and Technology Management, 2019.
[3] W. Vuorimaa, "Deep Learning Object Detection Models in Robotic Process Automation," Aalto University, December 2020.

[4] [Online]. Available: https://docs.uipath.com/ai-computer-vision/automation-suite/2023.4/user-guide/introduction. [Accessed May 2025].
[5] [Online]. Available: https://docs.uipath.com/activities/other/latest/ui-automation/computer-vision-activities. [Accessed May 2025].