Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2020.010124
Article

Video Preview Generation for Interactive Educational Digital Resources Based on the GUI Traversal

Xiaohong Shi[1,3], Chao Ma[2], Yongsheng Rao[3,*], Xiangping Chen[4] and Jingzhong Zhang[3]

[1]School of Mathematics and Information Science, Guangzhou University, Guangzhou, 510006, China
[2]National Engineering Research Center of Digital Life, Sun Yat-sen University, Guangzhou, 510006, China
[3]Institute of Computing Science and Technology, Guangzhou University, Guangzhou, 510006, China
[4]Guangdong Key Laboratory for Big Data Analysis and Simulation of Public Opinion, School of Communication and Design, Sun Yat-sen University, Guangzhou, 510006, China
[*]Corresponding Author: Yongsheng Rao. Email: rysheng@gzhu.edu.cn

Abstract: The interactive educational digital resources (IEDRs) are more and more prevalent in all levels of education. With the proliferation of the Internet, more and more IEDRs are being shared online. How to aid users to explore helpful resources effectively and efficiently has become a challenge. The static preview of resources that current research has mainly focused on is ineffective for the IEDRs, because of the unique feature of the IEDRs that the knowledge is crucial for the users' comprehension and is hidden in the process of interaction. To unfold the hidden crucial knowledge and ensure a users' fast acquisition, we proposed a novel approach to generate video previews for the IEDRs, which has achieved a better visualization of the IEDRs. First, the interactive element set and the events set, as well as the dependencies between the interactive elements of the resource are collected. Afterwards, by analyzing the dependencies, the interaction sequence is generated. Consequently, by automatically executing the interaction sequence based on the GUI traversal, the manual interaction is simulated, and the process is recorded into a video. The approach was applied into a dynamic mathematics educational system, SuperSketchpad, and its effectiveness was validated by a user study, in which it was compared with the random event method and event permutation method in correctness, completeness and redundancy.

Keywords: Interactive educational digital resource; video preview; GUI traversal; visualization

1 Introduction

With the proliferation of the Internet and soaring of online resources, there comes a challenging issue in helping users to efficiently discriminate helpful resources among such a slew of search results. Without previewing, users download needed resources, and even install specific tools for browsing, to identify whether they are helpful or not. If the resources are not as expected, the user may need not only to delete them but also to uninstall the tools. As a result, users waste their time on such repeated and unfruitful operations, which can make them feel frustrated in learning.

The resource preview, a means to address this issue and improve a users' experience, is becoming a research hotspot. The present resource preview methods mainly focus on a static exhibition of resources like webpages, image collections, videos, and audios. A study on the preview of interactive educational

digital resources (IEDRs), however, is deficient. It is complicated to generate previews for the IEDRs, because of the dynamic and interactive features.

With flourishing online education, IEDRs normally are produced by specific tools, such as SuperSketchpad (SSP) [1] and GeoGebra, which stand out among various resources for the effectiveness in enhancing a users' comprehension by interaction.

The increase of the IEDRs comes with several challenges for a users' exploration. The main challenge pertains to the comprehensibility of the IEDRs. Showing a lack of online previews for the IEDRs leads to a users' inconvenience in evaluating the helpfulness of the resources. Different from ordinary resources, most significant information in the IEDRs is hidden inside the interaction. Users need to interact with the IEDRs themselves by operating the elements (e.g., buttons) in the users' interface to acquire the hidden information, and a user cannot fully understand the IEDRs until accessing all the interactions. From a users' perspective, a straightforward and complete but brief preview of the resource is preferred rather than manipulation. The previews can help them to interpret and select resources more efficiently. From a resource constructers' perspective with a good preview, they can concentrate on the IEDRs design to deliver the knowledge with necessary interaction, and have no worry any more about that the users may be impatient to manipulate and comprehend the IEDR. In a further sense, the automatic preview generation for the IEDRs may benefit the development of some disciplines (e.g., dynamic geometry). We can summarize the design patterns from the popular IEDRs to guide a better resource development. Therefore, this research aims at proposing an approach to generate short video previews for the IEDRs, which will provide users with a clear picture of the IEDRs without any interaction with them.

The main contributions of this research include three aspects: 1) Proposing an approach to automatically generate video previews for the IEDRs, enabling users to perceive the contents of the IEDRs effectively and efficiently; 2) Applying the approach into the resources produced by the SSP; 3) Validating the efficacy of the proposed approach by experiments and user studies.

The remainder of the paper is organized as follows: Related work on the resource preview is introduced in Section 2. The proposed approach is detailed in Section 3. The application in the SSP is elaborated in Section 4, showing the feasibility of the approach. Experimental results and evaluation of the proposed approach are analyzed in Section 5, and conclusion and lines for future work are made in Section 6.

2 Related Work

We focus on providing users with previews for the IEDRs. From previews, users can acquire an easier but more complete understanding of online resources before they decide to download the resources and install the specific tools. In the following, we reviewed the existing related work from the three facets; preview methods for various kinds of resources, visualization technologies utilized and interactive graphical user interface (GUI) technologies.

A number of preview methods for videos [2–5], audios [6–7], webpages [8–10] and educational resources [11–12] have been proposed. A framework for quickly and interactively previewing digital contents with a semantic decomposition of complex multimedia contents, emulating "leafing the pages of an illustrated magazine" was proposed by [2]. But how to find a right decomposition is a problem for this framework. Luo et al. [3] presented a system to browse a collection of videos with three steps of an importance portion extraction, collage layout, and canvas rendering. This system is only applicable to videos. As videos are counting for more and more important parts in online resources, some techniques are required to aid users in efficiently finding the wanted videos. Craggs et al. [4] presented a method named ThumbReels, considering the tag data collected from viewers along with the traditional meta-data, creating the query-sensitive preview of the video. It is different from the contemporary preview techniques, focusing on the thumbnails and author-defined metadata. The preview, to some extent, is aware of the users' intension, narrowing the intention gap but relies too much on tagging that makes it less effective if the tagging accuracy is poor or the tags are inadequate. Yuan et al. [5] proposed a novel

graph convolved video thumbnail pointer (GTP), that not only provided a concise preview of the original video but also semantically corresponded to the given sentence description. These above-mentioned approaches are effective to videos, audios, or webpages. Because there are no interactions in videos and audios, and the interaction sequence is affected little on the webpage, these approaches do not fully consider, or consider the interaction sequence. Taylor et al. [11] proposed the RedFeather, a micro-repository, as a solution for open educational resource exhibition and discovery, including an in-line preview as one of its functions. For the IEDRs, a summarized preview generating method was proposed [12], simulating a users' interaction and recording snapshots step by step. Resource summaries provided by both methods of Taylor and Ma are static.

Visualization technologies are utilized to present the contents and output the previews of the resources. AutoCollage, an automatic procedure for constructing a collage from a set of input images was defined by [13]. It uses a multistage optimization procedure to tackle computational complexity and explicit a region of interest selection. The limitation of the AutoCollage is that it is not sensitive to a users' interactivity. For time-lapse videos, Gutwin et al. [14] improved the navigation from scrubbing with a slider to a frame-loading technique called spread-loading that enables scrubbing regardless of the delivery rate. Ahn et al. [15] presented an adaptive visualization method supporting navigation through class materials according to the lecture topics. In SIGGRAPH Asia 2017, Zhao et al. [16] proposed a method to automatically generate visual-textual web video thumbnails, providing an efficient way to preview video contents. Furthermore, Zhao et al. [17] designed and implemented a system for visual navigation of educational videos using multimodal cues, which integrates multimodal cues acquired from the visual, audio and textual channels of the video and presents them with a collection of interactive visualization components. DynamicSlide [18] improved the learner's video browsing and navigation experience by incorporating a set of reference-based techniques.

To simulate the automatic interaction in the IEDRs, the ideas of a traversal-based automated software testing can be made use of. The traversal-based automated software testing, from a users' perspective, by simulating manual execution, tests an application via its graphical user interface (GUI). The automated GUI testing has been approached and different techniques have been proposed [19]. Memon et al. [20] presented a new coverage criterion based on events in the GUI. The interactions between the events in a component were represented by an event-flow graph, a 4-tuple $<V, E, B, I>$, where V is a set of vertices representing all the events in a component, E a set of directed edges between vertices, B the events available to the user in the first invoking the component, and I the set of restricted-focus events of the component. The presented criteria can help determine whether a GUI has been adequately tested. For interactive resources, users usually perform interaction by executing events on their interfaces. Wang et al. [21] proposed an automatic android GUI traversal with high coverage, addressing the challenges in an automatic exploration for the GUIs of the Android app. Esparcia-Alcázar et al. [22] proposed to use the genetic programming (GP) to decide the action sequence that was defined as a list of IF-THEN rules. Kilincceker et al. [23] proposed an approach to randomly generate test sequences, by modelling the GUI by a finite state machine (FSM) and converting it to a regular expression. Likewise, to ensure all interaction events in the IEDRs are being executed at least once, the GUI traversal can be an effective solution. The IEDRs interactively illustrate knowledge by the interactive elements in them. The preview of an IEDR based on the GUI traversal with an inadequate interaction sequence may be hard to understand. There are dependencies between the interactive elements. An interactive element or a set of interactive elements may allow different sets of the operation. Therefore, in terms of the IEDRs, the dependencies are significant for generating an adequate interaction sequence. Considering this, we extracted the dependencies between the interactive elements, and leveraged them to generate the interaction sequence for the GUI traversal.

3 Video Preview Generation for IEDRs Based on GUI Traversal

The IEDRs differ from ordinary resources in the unique feature of interactivity, which cannot be browsed or previewed by conventional methods. Thus, focusing on the interactivity, we proposed a novel

approach to automatically generate video previews for the IEDRs. The workflow of the approach is illustrated in Fig. 1.

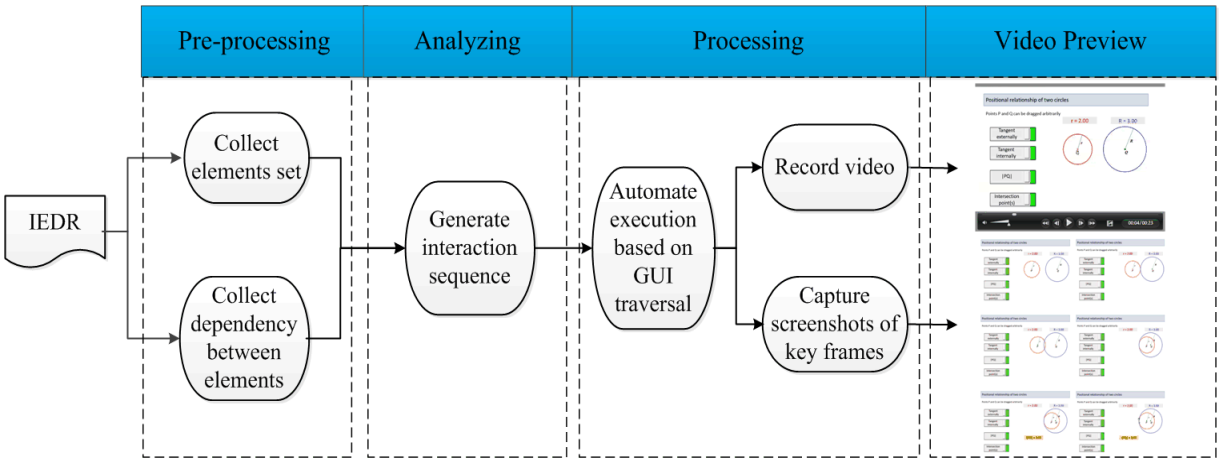


Figure 1: Workflow of video preview generation for the IEDRs

In the pre-processing stage, dependencies between the interactive elements are collected from the resource. Afterwards the interaction sequence is generated by analyzing the dependencies between the interactive elements. Executing the interaction sequence with some software automation tool, the user interaction is simulated, then with the help of some screen capture software the process is recorded into a video. Here we use the AutoHotkey for automatic execution and Snagit for screen capturing. Apart from the video, instantaneous screenshots of the interaction can be captured as the supplement to assist users in understanding the resource.

3.1 Data Pre-Processing

The interactivity of the IEDRs is fulfilled by the interactive elements. Before generating the preview, the structured description of the IEDRs and the dependencies between the interactive elements need to be collected.

3.1.1 Structured Description of the IEDRs

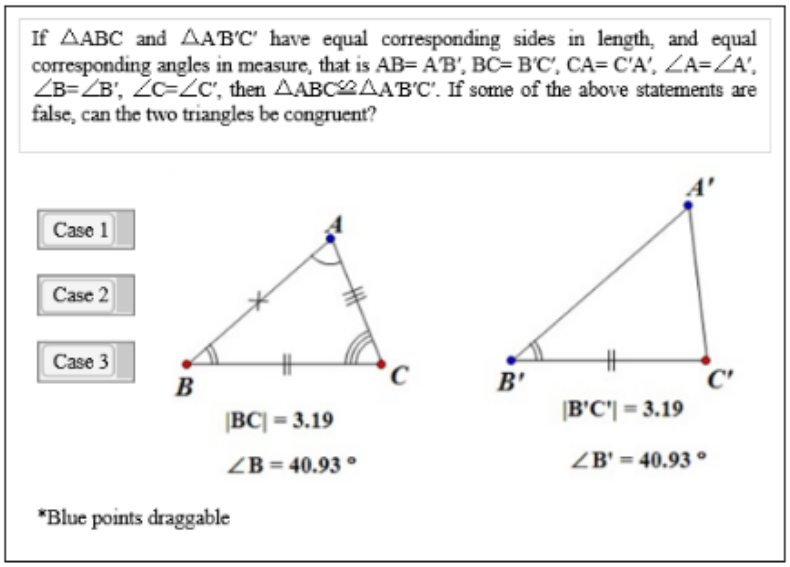


Figure 2: An instance of IEDRs. This is a screenshot of the IEDR "Congruent triangles" in SuperSketchpad

Ahead of detailing in the structured description, we brief the IEDRs with a specific instance. Fig. 2 displays a mathematics resource about learning congruent triangles, produced by SuperSketchpad [1], a dynamic mathematics educational system.

As seen from Fig. 2, there are texts, buttons, and graphs in this resource. Clicking the buttons or dragging the vertex will change the shapes of the triangles, for the purpose of enhancing the users' knowledge by observing the changes during the interaction. Here the buttons are interactive elements.

In the following we will make the structured description on an ordinary IEDR. An IEDR may contain one or multiple pages. For that with multiple pages, the page sequence is fixed. A structured description of an IEDR is defined as Eq. (1).

$$IR = (P, R), \tag{1}$$

where P is a set referring to all pages of the IEDR, and R relations between pages. A page is defined as Eq. (2).

$$p = (E, DR), \tag{2}$$

where $E = \{e_1, e_2, ..., e_n\}$ refers to all elements in the page, and DR refers to dependencies between the elements as defined in Eq. (3).

$$DR = \{< e_i, e_j > | e_i \in E \wedge e_j \in E\}. \tag{3}$$

An element in the page is defined as Eq. (4)

$$e =< id, eType, data, actionType, scope, reactionNum >, \tag{4}$$

where *id* is the identifier of the page; *data* refers to the data related to the element; *eType*, with values of *invisible*, *visible & interactive*, and *visible & static*, depicts the interactivity and visibility of the elements; *actionType* is the operation type with the input devices (e.g., keyboard, mouse); *scope* defines where the action can be performed in the interface; *reactionNum* is the number of element reactions according to the users' operation.

3.1.2 Extraction of Dependencies between the Interactive Elements

The possible dependency between the interactive elements in an IEDR is defined as a data dependency (DD) and control dependency (CD).

Algorithm 1 Pre-processing (E, DR)

Require:
elements set *E*; dependencies set *DR*
Ensure:
DR;
1: **for** $< e_i, e_j > \in DR$ **do**
2: **if** $e_j.eType \neq visible\&interactive$ **then**
3: **for** $e_k \in \{e | e \in E \wedge < e_j, \text{e} > \in DR\}$ **do**
4: **if** type$(< e_i, e_j >) == CD$ **then**
5: $DR = DR \cup \{< e_i, e_j >\}$;
6: **end if**
7: **end for**
8: **end if**
9: **if** $e_j.eType = visible\&static \wedge$ type$(< e_i, e_j >) ==$ DD **then**
10: **for** $e_k \in \{e | e \in E \wedge < e_i, \text{e} > \in DR\}$ **do**
11: **if** $type(< e_j, e_k >) ==$ type$(< e_k, e_j >)$ and $type(< e_k, e_j >) ==$ DD **then**
12: $DR = DR \cup \{< e_j, e_k >, < e_k, e_j >\}$;

13: **end if**
14: **end for**
15: **end if**
16: **end for**
17: $DR = \{< e_i, e_j > \mid < e_i, e_j > \in DR \wedge e_i.eType = visible\&interactive \wedge e_j.eType = visible\&interactive\}$;
18: $E = \{e | e \in E \wedge e.eType = visible\&interactive\}$;
19: **return** *DR*;

The data dependency (DD): If two elements are of DD, it means they are data associated. If action on element e_i depends on or changes the data of element e_j, the dependency between the two elements is DD. For example; given a line segment in an IEDR, containing two points, the operation on the line segment depends on the data of the two points. For another example, given a variable button in an IEDR, when clicking the button, the variable corresponding to the button varies, causing that value of the expression containing the variable to vary, and even further that the graph shaped by the expression changes.

The control dependency (CD): If element e_i is of the CD on element e_j, it means element e_i must be executed before element e_j. For example; given a button with the function of moving the position of a point that is also interactive and movable, the point's position will be changed as the button is executed, which will cause failure in executing the original point for some purpose, thus the point should be executed before the button.

The description of an IEDR can be collected based on the API interface information provided by their producing tools, including texts, events associated with interactive elements, as well as the dependencies between the elements. Algorithm 1 describes the pre-processing of the dependencies. Line 2 to 8 are to collect the dependencies between the current element and the elements of the CD. Line 9 to 15 are to collect the dependencies between the elements associated with the current element in the DD. Line 17 and 8 are to filter the dependencies associated with the non-interactive elements so as to reduce the number of dependencies. Our method generates the preview without the dependencies but with them the preview performance is boosted. We will validate this in the experiments.

3.2 Interaction Sequence Generation

By analyzing the collected dependencies, an interaction sequence can be generated. Afterwards automatically executing the sequence based on the GUI traversal, the interaction process is recorded into a video.

The interaction sequence is an order based on some traversal rule to interact with the GUI elements. It is composed of events triggered by the users from the interface. Such an event is defined as Eq. (5).

$$event =< eid, actionType, position >, \tag{5}$$

where *eid* indicates the identifier of the event, *actionType* indicates the type of action, and *position* indicates where the action is performed. An event is one interaction with the interactive element.

For any *event*, there exists an element *e* with an action scope including the event's position, having the same id, visible and interactive, and action type with it. The definition implicates that the invisible and non-interactive elements will not appear in the interaction sequence.

To boost the performance of the preview, we set the following two rules: 1) All the events associated with the interactive elements are executed at least once; 2) As many e_i and e_j's as possible are executed if <e_i, e_j> in *DR*. In addition, we make a hypothesis that the preview performance becomes lowered with the length of the interaction sequence increasing, which is verified in the experiments.

To fulfill rule 2, events may be executed more than once, which causes a redundancy of events and length increase of the interaction sequence. There obviously exists a conflict between rule 2 and the hypothesis. To tackle this conflict, two constraints are introduced; one is a cost function defined in Eq. (6) and Eq. (7); the other is a threshold that is the set for the number of redundant events.

$$Score = stepCost + opScor$$
$$= -0.5 + \sum_{0 \le k < L} opScore < S[k], S[k+1] > \quad (6)$$

$$opScore < e_i, e_j >= \begin{cases} 1, < e_i, e_j > \in DR, Times(e_i) = 1, Times(e_j) = 0; \\ 0, < e_i, e_j > \in DR, Times(e_i) > 1, Times(e_j) = 0; \\ -0.5, < e_i, e_j > \in DR, Times(e_i) = 1, Times(e_j) > 0; \\ 1, < e_i, e_j > \notin DR; \\ -2, < e_i, e_j > \in DR, Times(e_i) > 1, Times(e_j) > 0. \end{cases} \quad (7)$$

In Eq. (6) and Eq. (7), S is the sequence, L the length of the sequence, $S[k]$ event in the k^{th} step, $opScore<e_i, e_j>$, which is the score of interaction from event e_i to e_j , stepCost is the cost of all steps in the sequence (here stepCost is empirically set to -0.5, because it performs the best in most of our experiments), Times(e_i) is the number of occurrences of event e_i in S, telling if e_i is redundant or not.

Algorithm 2 depicts the generation of the interaction sequence. Line 5 to 16 are to select the events with the highest score as the next event. Lines 17 to 23 are to ensure the number of redundant events is less than the threshold. Lines 24 to 28 are to ensure each event being visited at least once and to decide to go on to visit the events or not if each event is visited and the number of redundancies is within the threshold. Here M is the threshold, that is the maximum number of the allowed redundant events, N the number of events, and DR is the dependencies set.

Algorithm 2 Generate Execution Sequence (e_i. *Events*, *DR, S, N, M, cost, counter*)

Require:

current event e_i; events set *Events*; dependencies set *DR*; interaction sequence *S*; number of events *N*; threshold of the number of redundant events *M*; cost of executing an event *cost*; counting number of redundant events *counter*; number of occurrences of event e_i in *S Times*();

Ensure:

interaction sequence *S*;

1: **if** *S.length* ≥ *N* + *M* **then**
2: **return** *S*;
3: **end if**
4: *times*1 = *Times*(e_i)
5: **for** *each* e_j *in Events* **do**
6: *times*2 = *Times*(e_j)
7: **if** $< e_i, e_j > \in DR$ **then**
8: $opScore[e_i][e_j] = 1$, *if times*1 == 1 *and time*2 == 0
9: $opScore[e_i][e_j] = 0$, *if times*1 > 1 *and time*2 == 0
10: $opScore[e_i][e_j] = -0.5$, *if times*1 == 1 *and time*2 > 0
11: $opScore[e_i][e_j] = -2$, *if others*
12: **else**
13: $opScore[e_i][e_j] = -1$
14: **end if**
15: **end for**
16: $e_j = \{e | e \in Events \wedge v \in Evens \wedge opScore[e_i][e] \ge opScore[e_i][v]\}$
17: **if** $opScore[e_i][e_j] == -0.5$ **then**
18: if *counter* > *M* then
19: $e_j = \{e | \exists e \in Events \wedge opScore[e_i][e_j] = -1\}$
20: **else**
21: *counter* + +
22: **end if**

```
23: end if
24: if each event is visited and counter ≤ M then
25:    if opScore[e_i][e_j]+ cost > 0 then
26:       S.append(e_j);
27:       S.score = S.score + opScore[e_i][e_j]+ cost
28:       GenerateExecutionSequence(e_j, Events, DR, S, N, M, cost, counter)
29:    else
30:       return S
31:    end if
32: end if
```

In Algorithm 2, with the *DR* increasing, the interaction increases, and the interaction sequence is more adequate. For threshold *M*, the bigger its value is, the longer the interaction sequence is. As the afore-made hypothesis, the increasing length of the interaction sequence will lower the preview performance. We set $M = N/3$ in the experiments.

3.3 The GUI Traversal on the IEDRs and the Preview Generation

Our focus is on how to generate a sequence for the GUI traversal, which is detailed in Section 3.2. With the interaction sequence, we realized the GUI traversal by employing the AutoHotKey, an open-source custom scripting language for providing software automation. The whole interaction with the IEDRs can be recorded by some screen recording software into a video. In our work, we use the Snagit. During the traversal, the instantaneous screenshots of every interaction can also be captured by the Snagit. We will make it more specific to the application in Section 4.

By viewing the video, users can acquire how to interact with the resource and what is provided in the resource. However, sometimes users may just hope to get familiar with some important contents of the resource, instead of the entire resource. Under such a condition, the instantaneous screenshots serve as alternatives.

4 Application in SuperSketchpad

To show the feasibility of the video preview generation method, it was applied into the IEDRs from the SuperSketchpad (SSP).

Before the automating interaction with the IEDR in the SSP, the interactive elements in the IEDR need to be collected. The IEDR in the SSP can be saved as html files, from which the interactive elements can be identified.

4.1 Categorization of the Interactive Elements

Interactive elements of the IEDRs in the SSP can be recognized from the graphic functions in the function library documents. We simulated the execution of about 100 graphic functions from the 114 randomly selected IEDRs in the SSP and identified the interactive elements and their corresponding element names in the resource files by the following steps.

Step 1: Open the resource and save it as a basePage without any operation.

Step 2: After execution of an element in the resource, save the post-operation resource as a changedPage.

Step 3: Identify the corresponding element name in the resource files by finding the difference between the basePage and the changedPage with the Beyond Compare (from http://www.scootersoftware.com/).

Following the steps above, with the aid of the descriptions about the graphic functions in the function library documents, we get the interactive elements, as categorized in Tab. 1. There are 6 categories and 11 types of interactive elements in total.

The complex animation button here refers to the button that needs clicking at least twice to complete the execution. For example, with one click, the animation in the resource starts, while with another click, it stops.

4.2 The Position Calculation of the Interactive Elements

The information in Tab. 1 is insufficient, so the specific positions of the elements in the IEDR are also required. For a simple animation button, by *ObjPosition (nObjID, nLeft, nTop, nRight, nBottom)*, in the resource file, its position can be acquired, that is *(nLeft, nTop, nRight, nBottom)*.

The acquired position of an element is an absolute position, but the element automatically executed by the AutoHotKey is in the SSP, which means it cannot be located directly with the acquired position. Actually, the element's position in the SSP should be defined by the obtained position plus its offset in the SSP.

Table 1: Interactive elements

Category	Description	Name of element
Simple animation button	Need one click only	AnimationVar. AnimationParallel AnimationSerial Button
Complex animation button	Need at least two clicks	Animation
Variable bar	Its value can be changed by dragging the slider within the range on the bar	Variable
Normal point	Can be dragged along x-axis, y-axis or both	Point
Complex point	Represent the complex number of which x-coordinate is the real part, y-coordinate is the imaginary part	ComplexPoint
Point on lines	Can be the point of a conic, a straight line or a perpendicular line to a straight line	PointOnConic PointOnLine PointOnVLine

Randomly selecting dozens of resources from the SSP via the statistical analysis, we got the average coordinates offset of the animation buttons that is (172,114).

In fact, by calculating the button's coordinates in the html file plus its corresponding coordinate offset, what we got is the coordinates of its upper left vertex. Sometimes deviation might appear, which would cause no response when clicking the button, thus we set the coordinate offset as its value plus 4 both in the x-axis and y-axis, so as to avoid the deviation, then we got the offset of (176, 118).

We repeated the process above to calculate the coordinate offsets and got the coordinates from the resource file for different types of interactive elements in Tab. 1. Then the actual coordinates of all interactive elements were acquired with the absolute coordinates plus the offset.

4.3 Video Preview

Based on the data prepared in Sections 4.1 and 4.2, as well as the method detailed in Section 3, the AutoHotKey scripts can be written to automate the preview generation process. Through the AutoHotKey, the SSP are opened then the IEDR is opened. After that, the interactive elements in the IEDR are identified and their positions are calculated as described in Sections 4.1 and 4.2. With the method detailed in Section 3, the interaction sequence for the IEDR is generated. Consequently, the GUI traversal is automated by the AutoHotKey following the interaction sequence and recorded by the Snagit into a video.

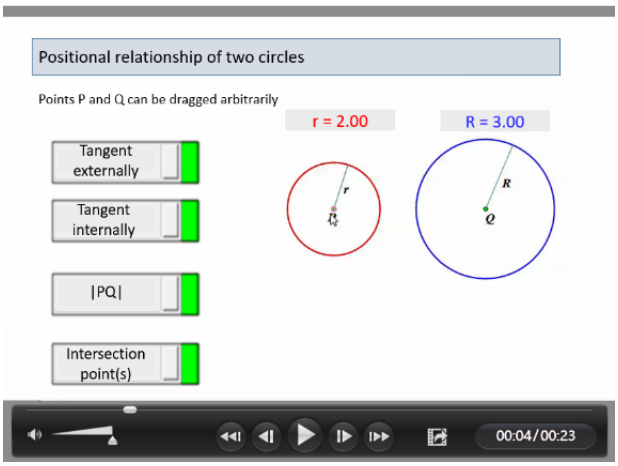
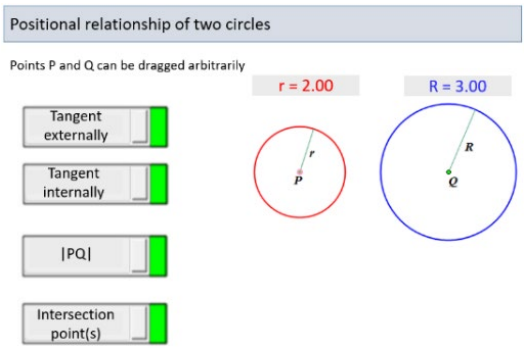
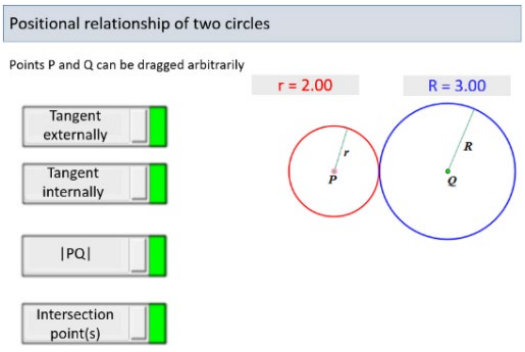


Figure 3: A screenshot of the video preview of an IEDR in the SSP

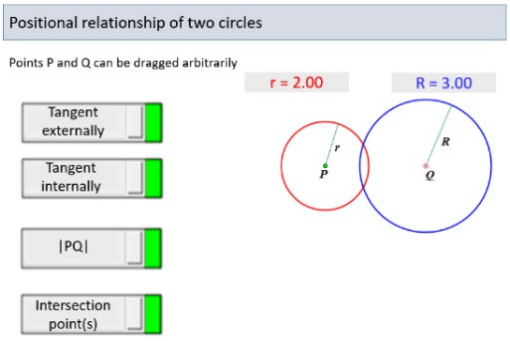
During the traversal, the instantaneous screenshots of every interaction are captured as well. The IEDRs can be processed and traversed in a batch. We took one of the resources as an example. Fig. 3 is a screenshot of the video. Fig. 4 shows the instantaneous screenshots captured as the supplementary with the GUI traversal on the interaction sequence of the resource, 6 screenshots being captured as 6 events were executed. From the screenshots, users get to know what interaction is included in the IEDR and roughly acquire the knowledge by observing the changes, which will be complementary for briefing the video preview.



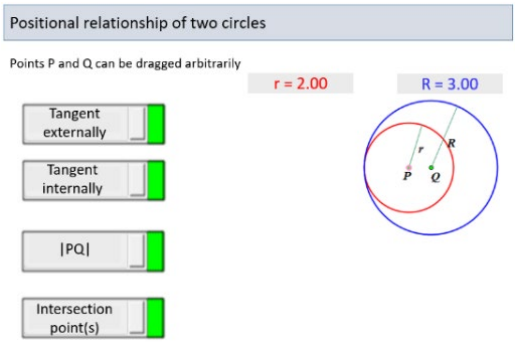
(a) Drag Point P



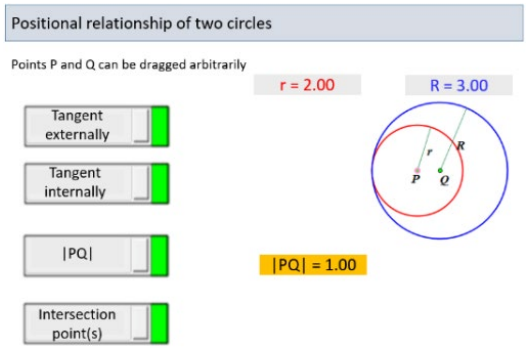
(b) Click the button "Tangent externally"



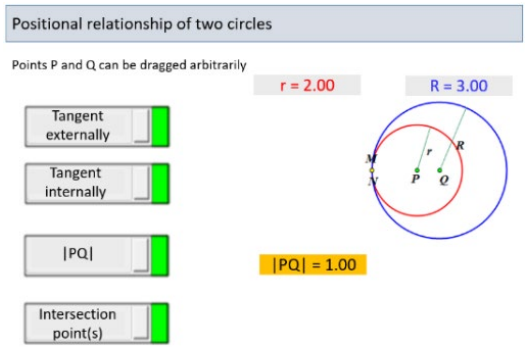
(c) Drag Point Q



(d) Click the button "Tangent internally"



(e) Click the button "|PQ|"



(f) Click the button "Intersection point(s)"

Figure 4: Instantaneous screenshots captured when the GUI traversal on the interaction sequence of the resource

5 Experiments

The proposed video preview generation method has been applied into the IEDRs in the SSP. Meanwhile a resource retrieval platform based on the method has been implemented, providing effective service for the students of Guangzhou University in the courses associated with the SSP.

To verify the effectiveness of the proposed method, we designed a questionnaire and invited 65 participants to participate in the evaluation, among whom 60 are from Guangzhou University and 5 are from Sun Yat-sen University. All of them had studied a course about the SuperSketchPad.

We randomly selected 500 resources from the SSP. With considering the resources with fewer than three events being too simple and not meaningful enough for the evaluation, we filtered them out and finally received 367 resources. We thought 10 resources in one questionnaire was an appropriate number. As a result, we first made a random selection of 360 resources from the 367 resources, and separated them into 36 groups, each containing 10 resources; then we randomly picked 3 from the grouped 360 resources to make another group with the remaining 7 resources. Finally, we received 37 groups in total, all containing 10 resources.

For comparison, two baseline methods were implemented on the selected IEDRs to generate the video previews. To our knowledge, there is no research on video preview generation for the IEDRs until now, thus we must compare our method with the baseline methods. One is a GUI traversal method based on the random event sequence (hereafter called random method), not considering the dependencies between the elements; the other is a GUI traversal method based on the concatenating sequence of permutations of events in the resource (not necessary to consider all the permutations, here the first three permutations are selected and concatenated for traversal, hereafter called the permutation method). Obviously, the sequence length of the permutation method is the longest. The reason for comparison with the permutation method is to verify the negative effect of the increasing sequence length on the preview performance, which is the hypothesis made in Section 3.2.

Table 2: Data collected from the questionnaires

Questionnaire	Resource	Correctness			Completeness			Redundancy		
		P1	P2	P3	P1	P2	P3	P1	P2	P3
Q1	R1	3	4	5	2	4	5	2	3	2
	R2	4	5	4	2	5	4	2	1	4
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	R10	3	3	4	2	1	4	1	1	2
Q2	R1	4	4	3	5	3	3	3	2	2
	R2	4	5	4	4	5	4	2	3	4

	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	R10	4	4	3	5	4	4	2	4	3
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	R1	4	5	3	3	4	5	1	1	4
	R2	3	4	5	4	5	3	1	1	4
Q39	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	R10	4	5	3	5	5	4	1	1	3
	Average	3.84	**4.07**	3.97	3.57	**3.94**	3.89	2.00	**2.11**	2.86

[1] P1, P2 and P3 refer to the video previews generated by the random method, the proposed method, and the permutation method, respectively. R1, R2… and R10 refer to the resources rated in the corresponding questionnaire, respectively.

[2] The resources in each questionnaire depend on the assignment among the 37 resource groups.

Based on the questionnaires received, we filtered the questionnaire if it was (1) not completely answered; (2) not correctly answered (e.g., answers of some questionnaires rating the same group of resources are highly similar). We finally received 39 valid questionnaires, in which the video previews for 240 IEDRs were rated. Among these resources, 164 (i.e., about 68%) have at least one dependency and up to 18 dependencies, while 76 (i.e., about 32%) have no dependencies. Before the participants rated the video previews, they were asked to install the SSP, and open the resources with it, interact with the resources and try to understand the contents. The understanding based on their interaction can help them compare the quality of the video previews generated by the three methods. Part of the data from the valid questionnaires are illustrated in Tab. 2.

The quality is evaluated from correctness, completeness and redundancy of interaction, all scores in the integer ranging from 1 to 5. We clarified to the participants that 1) Correctness indicates interaction in the video preview is identical to that in the corresponding source; 2) Completeness indicates all interaction in the source is carried out in the video preview; and 3) Redundancy indicates duplicated interaction exists in the video preview. The participants were asked to score and vote the best one among the three video previews generated by these three methods from the same resource. To avoid biases, the participants were not told by which method the preview was generated.

For the scores of correctness and completeness, the higher is better. The average correctness on the video previews by the proposed method scored 4.07, while by those by the random method and permutation method 3.84 and 3.97, respectively. The scores of completeness on the video previews by the proposed method averaged 3.94, while those by the random method and permutation method 3.57 and 3.89, respectively. Although 32% of the rated IEDRs have no dependency, our proposed method still averagely outscores the other two methods both in correctness and in completeness, which verifies its effectiveness in generating the previews for the IEDRs either with dependencies or without dependencies. Moreover, it also suggests that considering dependencies between the interactive elements in generating the traversal sequence boosts the performance of the video previews.

In terms of redundancy, the lower the score is, the better the method performs. The permutation method is the worst among the three methods, with an average score of 2.86. The result reflects that the increasing length of the sequence will lower the performance in correctness and completeness, that is, the hypothesis is verified. The proposed method achieved an average score of 2.11, comparable to the random method (2.00), which suggests that the proposed method can effectively control the redundancy by setting the constraints.

Furthermore, the comparison about the distribution of the scores each method received, illustrated in Tab. 3, also suggests that the proposed method has an overall advantage over the other two methods. Since the score ranges from 1 to 5, the comparison is made in three parts, the mean score 3, lower scores of 1 and 2, and higher scores of 4 and 5. The proposed method tops the other two methods in higher scores in correctness and completeness, with 76% and 70%, respectively. In terms of redundancy, 70%

scoring of the random method are of 1 or 2, just showing a very weak superiority over the proposed method (67%). It indicates that allowing the events to be executed more than once so as to fulfill rule 2 in Section 3.2 does cause redundancy but the proposed method can reduce this negative effect and approximate the random method to a great extent by setting the constraints.

However, the correctness and completeness are the priors and of great significance for the users' comprehension, which makes the tiny sacrifice in redundancy well worth it.

To give an intuitive demonstration of the experimental results, Fig. 5 exhibits the correctness scores of the video previews for 15 resources randomly out of the 367 resources.

According to the statistics from the 39 valid questionnaires, 48.33% of the best-voted previews are generated by the proposed method, far leading the compared methods, the random method (22.62%) and the permutation method (29.05%).

Table 3: Score distribution of the three methods

Valuation	Method	Scores		
		4 or 5	3	1 or 2
Correctness	Random Method	64%	31%	5%
	Proposed Method	**76%**	21%	3%
	Permutation Method	72%	23%	5%
Completeness	Random Method	52%	36%	13%
	Proposed Method	**70%**	24%	5%
	Permutation Method	67%	26%	7%
Redundancy	Random Method	11%	19%	**70%**
	Proposed Method	14%	19%	**67%**
	Permutation Method	32%	29%	38%

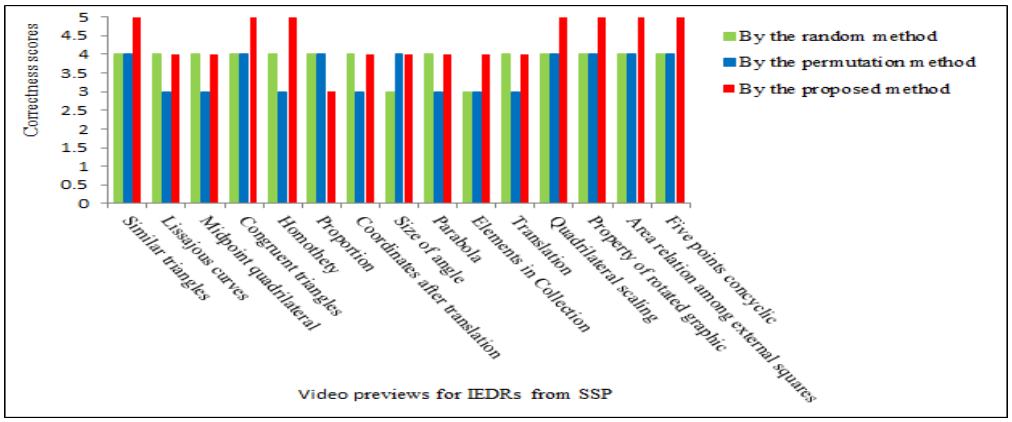


Figure 5: The correctness scores of the video previews

The distribution of best voted previews was influenced by the number of dependencies between the interactive elements. Fig. 6 depicts the relation between the percentage of best votes to the video previews generated by different methods and the number of dependencies in the corresponding resource.

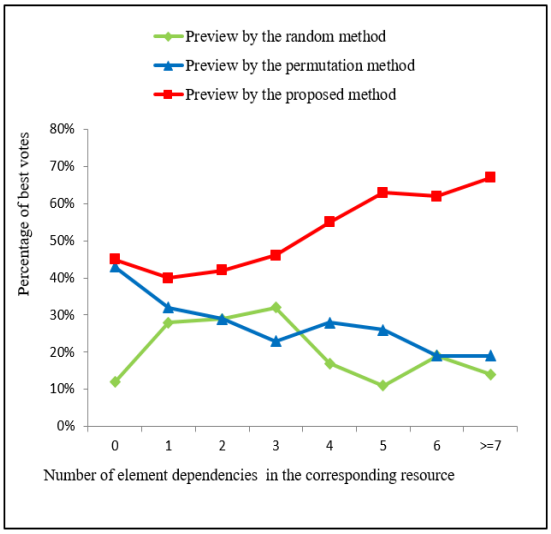


Figure 6: The relation between the percentage of the best votes to the video preview generated by the different methods and the number of element dependencies in the corresponding resource

As seen from Fig. 6, the percentage for the proposed method increases greatly with the increase of the number of dependencies. On the contrary, those for the other two methods fluctuate and have a dropping tendency, all much lower than the proposed method. The results are reasonable. If no dependency exists among elements (i.e., $DR = \{\}$), the interaction sequence has no obvious influence on the results of the proposed method and the permutation method. But as the number of dependencies increases, the influence of the interaction sequence becomes more and more obvious, and the results from different sequences are in a great difference, which is why more best-votes go to the results of our methods in the cases with more dependencies. The comparison between the random method and the proposed method suggests that considering the dependencies between the interactive elements in generating the interaction sequence provides better preview performance. The comparison between the permutation method and the proposed method verifies that the increasing length of the interaction sequence lowers the preview performance.

From the above analyses, we can see that the proposed method generally outperforms the compared methods in correctness, completeness, and redundancy as well as improving the quality of the preview in the IEDRs and providing users with better experiences.

6 Conclusion

To reduce a users' burden of downloading and manipulating the search results of the IDERs, we proposed a method based on the GUI traversal to automatically generate video previews that are straightforward and complete but brief. Before generating the preview, the structured description of the IEDRs was defined. The description, events and dependencies between the interactive elements are collected. Subsequently, by analyzing the dependencies the interaction sequence is generated. With the GUI traversal on the sequence is done by the AutoHotKey, the whole interaction is recorded into a video by Snagit. Compared with the random method and the permutation method, the proposed method leads in correctness and completeness, and its performance in redundancy is comparable to the random method. The proposed approach gives a better solution to the preview of the IEDRs and is expected to help the instructors and students acquire the educational resource of interest.

Nevertheless, it still has some limitations. During the GUI traversal, if the interaction with an element causes some other interactive elements' change in the *eType* to be non-interactive, interacting with these changed elements cannot be executed as a result. Some detection for the state change of elements during the GUI traversal should be considered to address this problem in the future. From the

technical perspective, consideration from the perspectives of the communication design as well as education is also meaningful for reducing complexity in finding the useful IEDRs. In addition, human-computer interaction assessments and educational assessments may be conducted to investigate the effectiveness of the proposed approach in the future.

Funding Statement: This work is partially supported by the National Key R&D Program of China (No. 2018YFB1005104), the Guangzhou Academician and Expert Workstation (No. 20200115-9), Key Disciplines of Guizhou Province–Computer Science and Technology (ZDXK [2018]007), Guangdong Basic and Applied Basic Research Foundation (2020A1515010973), the Fundamental Research Funds for the Central Universities and the Graduate Innovative Capacity Cultivation Program (2018GDJC-D03) of Guangzhou University.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Zhang and C. Li, "An introduction to logical animation," in *Int. Workshop on Mathematics Mechanization, Proc.: Lecture Notes in Computer Science*, vol. 3519, Springer, Berlin, Heidelberg, pp. 418–428, 2004.

[2] A. Barletta, M. Mayer and B. Moser, "Leafing digital content," in *Proc. of the 9th Int. Conf. on Intelligent User Interfaces*, Madeira, Funchal, Portugal, pp. 217–219, 2004.

[3] S. Luo, C. Tsai and H. You, "Previewing video content with dynamic and interactable collage," in *ACM SIGGRAPH Asia 2012 Posters*, Singapore, 2012.

[4] B. Craggs, M. K. Scott and J. Alexander, "Thumbreels: Query-sensitive web video previews based on temporal, crowd sourced, semantic tagging," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, Toronto, Canada, pp. 1217–1220, 2014.

[5] Y. Yuan, L. Ma and W. Zhu, "Sentence specified dynamic video thumbnail generation," in *Proc. of the 27th ACM Int. Conf. on Multimedia*, Nice, France, pp. 2332–2340, 2019.

[6] A. S. Shirazi, A. H. Sarjanoja and F. Alt, "Understanding the impact of abstracted audio preview of SMS," in *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, Atlanta, Georgia, USA, pp. 1735–1738, 2010.

[7] A. Lehtiniemi and J. Holm, "Easy access to recommendation playlists: Selecting music by exploring preview clips in album cover space," in *Proc. of the 10th Int. Conf. on Mobile and Ubiquitous Multimedia*, Beijing, China, pp. 94–99, 2011.

[8] P. Parente, "Audio enriched links: Webpage previews for blind users," in *Proc. of the 6th Int. ACM SIGACCESS Conf. on Computers and Accessibility*, Atlanta, Georgia, USA, pp. 2–8, 2004.

[9] B. Yoo, J. Lea and Y. Kim, "The seamless browser: Enhancing the speed of web browsing by zooming and preview thumbnails," in *Proc. of the 17th Int. Conf. on World Wide Web*, Beijing, China, pp. 1019–1020, 2008.

[10] A. Aula, R. M. Khan and Z. Guan, "A comparison of visual and textual page previews in judging the helpfulness of web pages," in *Proc. of the 19th Int. Conf. on World Wide Web*, Raleigh, North Carolina, USA, pp. 51–60, 2010.

[11] M. Taylor, Y. Howard and D. E. Millard, "Redfeather–resource exhibition and discovery: A lightweight micro-repository for resource sharing," in *Proc. of OER13: Creating a Virtuous Circle*, Nottingham, England, UK, pp. 1–8, 2013.

[12] C. Ma, X. Chen, Y. Rao and M. Lin, "Generating summarized preview for education resource based on exploring and comparing GUIs," in *Proc. of SEKE 2016*, Redwood City, San Francisco Bay, USA, pp. 141–146, 2016.

[13] C. Rother, L. Bordeaux and Y. Hamadi, "Autocollage," in *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 847–852, 2006.

[14] C. Gutwin, M. van der Kamp, M. S. Uddin, K. Stanley, I. Stavness *et al.,* "Improving early navigation in time-lapse video with spread-frame loading," in *Proc. of the 2019 CHI Conf. on Human Factors in Computing Systems*, Glasgow, Scotland, UK, pp. 1–12, 2019.

[15] J. Ahn and P. Brusilovsky, "Guiding educational resources for iSchool students with topic-based adaptive visualization," in *Proc. of the 2011 iConf.*, Seattle, USA, pp. 632–633, 2011.

[16] B. Zhao, S. Lin, X. Qi, Z. Zhang, X. Luo *et al.,* "Automatic generation of visual-textual web video thumbnail," in *ACM SIGGRAPH Asia 2017 Posters*, Bangkok, Thailand, 2017.

[17] B. Zhao, S. Lin, X. Luo, S. Xu and R. Wang, "A novel system for visual navigation of educational videos using multimodal cues," in *2017 ACM on Multimedia Conf.*, Mountain View, USA, pp. 1680–1688, 2017.

[18] H. Jung, H. V. Shin and J. Kim, "Dynamicslide: Reference-based interaction techniques for slide-based lecture videos," in *Proc. of the 31st Annual ACM Sym. on User Interface Software and Technology Adjunct Proceedings*, Berlin, Germany, pp. 23–25, 2018.

[19] A. Rauf and E. A. Aleisa, "PSO based automated test coverage analysis of event driven systems", *Intelligent Automation & Soft Computing*, vol. 21, no. 4, pp. 491–502, 2015.

[20] A. M. Memon, M. L. Soffa and M. E. Pollack, "Coverage criteria for GUI testing," in *Proc. of the 8th European Software Engineering Conf.*, Vienna, Austria, pp. 256–267, 2001.

[21] P. Wang, B. Liang and W. You, "Automatic android GUI traversal with high coverage," in *the Fourth Int. Conf. on Communication Systems & Network Technologies*, Bhopal, India, pp. 1161–1166, 2014.

[22] A. I. Esparcia-Alcázar, A. Francisco, T. E. J. Vos and U. Rueda, "Using genetic programming to evolve action selection rules in traversal-based automated software testing: Results obtained with the TESTAR tool," *Memetic Computing*, vol. 10, no. 3, pp. 257–265, 2018.

[23] O. Kilincceker, A. Silistre, M. Challenger and F. Belli, "Random test generation from regular expressions for graphical user interface (GUI) testing," in *2019 IEEE 19th Int. Conf. on Software Quality, Reliability and Security Companion*, Sofia, Bulgaria, pp. 170–176, 2019.