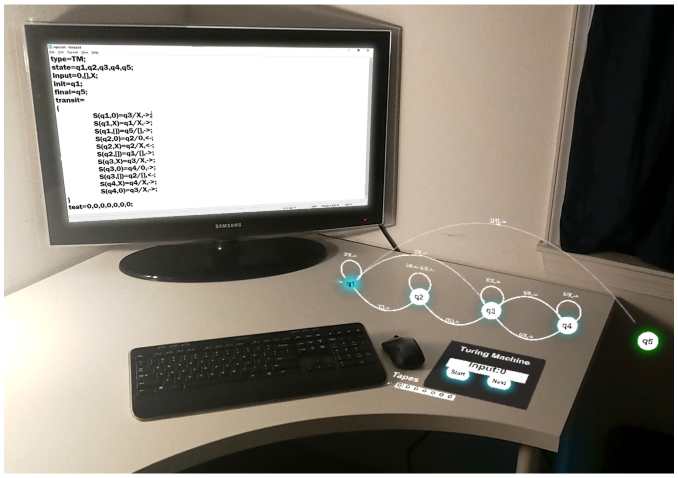
Simulating Turing Machine in Augmented Reality

Wanwan Li
University of South Florida
Tampa, Florida, USA

***Abstract*—We present an easy-use interactive user interface for simulating Automaton and Turing machine in Augmented Reality (AR). We design a C-like programming language to describe the automaton and Turing machine, visualize their data structures, and simulate their running process in AR given to the user-specified input string or tape. Our Automaton and Turing machine simulator support three different modes, including Finite Automata (FA), Pushdown Automata (PDA), and Turing Machine (TM). We have deployed our AR Turing machine simulator on HoloLens2, users can interact with the simulator through AR buttons at the same time they can edit the scripts on the desktop where there is a USB cable connecting the desktop and HoloLens2 such that the Turing machine simulator can be updated simultaneously as users update their scripts. This mechanism can enable the user to efficiently design the Turing machine through our interactive interface in augmented reality.**

***Index Terms*—augmented reality, automaton theory, Turing machine simulator human-computer interaction**



```
type=TM;
state=q1,q2,q3,q4,q5;
input=0,[],X;
init=q1;
final=q5;
transit=
{
        S(q1,0)=q3/X,->;
        S(q1,X)=q1/X,->;
        S(q1,[])=q5/[],->;
        S(q2,0)=q2/0,<-;
        S(q2,X)=q2/X,<-;
        S(q2,[])=q1/[],->;
        S(q3,X)=q3/X,->;
        S(q3,0)=q4/0,->;
        S(q3,[])=q2/[],<-;
        S(q4,X)=q4/X,->;
        S(q4,0)=q3/X,->;
}
test=0,0,0,0,0,0,0;
```

Fig. 1. AR Turing Machine Simulator.

I. Introduction

As one of the greatest cornerstones in computer science, Alan Turing proposed the first prototype of modern digital computers, namely, the Turing machine. As described by Turing himself [1], a computing machine can read symbols on tape and make the corresponding reactions including update its configuration or rewriting that input symbol as output. This epoch-making description of abstract machines influences several generations who are aiming at design computers and algorithms. Even today, for those people who are majored in computer science, computer scientists, and researchers, computational theories are still fundamental topics that require a solid background in Turing machine and automation theories. Therefore, for education purposes, tons of Turing machine simulators appear nowadays and be applied to the teaching of computation theory courses. It becomes a hot research topic to develop an effective user interface to help beginners understand the concepts about Turing machine more quickly.

However, most of the existing platforms for Turing machine simulations are not easy to learn as there are lots of complex operations and definitions that are not covered by the best-selling text book [2] of introduction for computation theory which is using an easy-to-use C-like description for Turing machines. As a consequence, it takes extra time for most beginners to get familiar with and master the skills of how to use that simulator. On the other hand, virtual reality and augmented reality technologies are getting popular, especially due to the VR and AR's impact on various types of educational programs such as creative arts [3], dancing [4], and diving [5] etc., virtual training [6] and interactive simulation also become a trend for modern computer science education due to their immersiveness [7]. Thusly, AR can be a widely welcomed platform to illustrate the Turing machine for effective training and over-perform other platforms.

Given these observations, by taking the advantages of the augmented reality technology and lowering the learning curve for the beginners through a novel C-like and easy-use programming script, we propose an immersive 3D interactive interface for Turing machine design and simulation in augmented reality as shown in Figure 1. After deploying our AR Turing machine simulator onto an AR device named HoloLens2, users can interact with the simulator through AR buttons. At the same time, there is a USB cable connecting the desktop and HoloLens2, therefore, users can edit the scripts on the desktop and the Turing machine simulator can be updated simultaneously when users update their scripts. This makes our system a plausible platform for the users to design their Turing machines with interactive interfaces in augmented reality. Video demo of our work can be found through this link: https://youtu.be/U1vLvY9YPwg

II. Related Work

In view of the importance of automaton theory and the great impact brought by the Turing machine, lots of successful works have been done to simulate automaton and Turing Machines. Even at an early age, Hennie et al. [8] proposed a two-tape simulation of multitape Turing machines. After then, Robson et al. [9] has moved forward to simulate a single tape Turing machine through a fast probabilistic random-access machine (RAM). Furthermore, Dauchet et al. [10] simulates the Turing machines by a left-linear rewrite rule and with a regular rewrite rule [11]. Later days, another great work has been done by Carpentieri et.al. [12] to simulate Quantum Turing Machine (QTM). After then, more follow-up works have been studied such as extending the Turing machine simulator with uEAC-Computable Functions proposed by Zhu et al. [13]. Recently, Dengel et al. [14] propose a metaphorical representations that converts different states in a finite state machine into a group of islands in virtual reality to improve the educational quality. Nowadays, virtual reality play more and more important roles in in computer science educations [15]. For example, VR has been used by Zable et al. [7] to develop an effective tool for quantum computing education. Also, convolutional neural networks (CNN) can be visualized with virtual reality [16] to help students to understand how CNN works during the training process.

However, nowadays, most Automaton and Turing machine simulators are implemented as desktop apps or command line-based apps. One famous work has been done by Hamada et al. [17], a desktop version app of PAD and TM simulator that helps students improve learning has been well-developed. On this platform, students have to create the automation by clicking buttons and drag arrows, therefore heavy mouse interactions are needed to design the machine. We have overcome such limitations by proposing an easy-to-use C-like script to describe a machine directly. Similarly, some other work on Automaton and Turing machine simulators such as [18], [19], and [20] have common drawbacks such as the learning curve is so high or the visualizations are either not straightforward or over complicated. For example, one online Turing machine simulator [18] has an over-complicated grammar to encode a Turing machine, so, it takes a longer time for students to master the skill to create a Turing machine to simulate. As another example, [19] simulates the Turing machine on the command line and it is very hard to read also is not user-friendly to use. Similarly, [19] shows another complex interface to simulate the Turing machine and there is a long tutorial needs to follow to create your own simulations.

Therefore, our work is to make an automaton and Turing machine simulator easy-to-learn and fun-to-use. We have to important notions that (1) our input script to simulate the Turing machine is very carefully following the textbook standards so that students are capable of making hands-on immediately after reading the textbook. (2) We have take the advantage of Augmented Reality technologies to make the learning process very impressive and attractive, as AR applications provide such impressiveness by their attractive natures. Given these two observations, in this paper, we proposed and introduce the first well-designed Augmented Reality-based simulator for automaton and Turing machine simulation.

III. Automaton Theory Fundamentals

The term ”Automaton” by definition is a ”self-acting unit”. An automaton (Automata in plural) is an abstract self-acting computing device that follows a sequence of operations automatically. Bellow are three types of automaton including Finite Automaton (FA), Push Down Automaton (PDA), and Turing Machine (TM). Finite Automaton (FA) is an automaton with a finite number of states is called a Finite Automaton (FA) or Finite State Machine (FSM). Pushdown Automaton (PDA) is a type of automaton that can accept context-free grammar. A FA can remember a finite amount of information, but a PDA can remember an infinite amount of information. Basically, a pushdown automaton is a finite automaton with a stack. In general, a pushdown automaton has three main components: (1)an input string, (2) a control unit, and (3) a stack with infinite size. The stack head scans the top symbol of the stack. A stack does two operations: (a) Push: a new symbol is added at the top of the stack. and (b) Pop: the top symbol in stack is read and removed. Turing Machine (TM) is an accepting device that accepts the recursively enumerable languages generated by type-0 grammars which were invented in 1936 by Alan Turing. A Turing Machine (TM) consists of an infinite length tape divided into cells where there is the given input. It consists of a head that reads or writes the tape and a state register. After reading an input symbol, that symbol might be replaced, its internal state is changed, and it moves from one cell to another by moving the head right (->) or left (<-). If the TM reaches the final state, the input string is accepted, otherwise rejected. In our system, both Finite Automaton (FA), Pushdown Automaton (PDA), and Turing Machine (TM) can be defined with C language-like grammar. As shown in Figure 2, in this example, the scripts describes a Turing machine (TM) that simulates a string copier that copy 1s on the tape. Then we will interpret such scripts though our interpreter and construct the data structure of such Turing Machine (TM) in both VR and AR for interactive simulations.

```
type=TM;                          % Specify the automaton type as TM
state=q0,q1,q2,q3;                % Specify different states
input=0,1,[],X;                   % Specify input alphabet
init=q0;                          % Specify initial state
final=q3;                         % Specify final state
transit=                          % Specify transition rules:
{
        S(q0,1)=q0/X,->;          % (Current State,Read)=Next State/Write, Move
        S(q0,[])=q1/[],<-;
        S(q1,1)=q1/1,<-;
        S(q1,X)=q2/1,->;
        S(q1,[])=q3/[],->;
        S(q2,1)=q2/1,->;
        S(q2,[])=q1/1,<-;
}
test=1,1,1,1,1;                   % Test input symbols
```

Fig. 2. One example of Turing Machine (TM) represented in our proposed C language-like scripts. This TM accepts all the language consists of only 1s. At the same time, it copy 1s on the tape, in the end, there will be doubled number of 1s appears on the tapes when the TM halts.

IV. Implementations

A. Data Structures

We developed a program that converts the easy-use C language-like scripts into Turning Machine’s data structure for

	q0	q1	q2	q3
q0	1/X,->;	[]/[],<-;		
q1		1/1,<-;	X/1,->;	[]/[],->;
q2		[]/1,<-;	1/1,->;	
q3				

Fig. 3. The transition table of a Turing machine.

interactive simulation. As defined in Sec III, those statements followed by the keywords such as "type=...", "state=...", and "final=..." specifying the structure of the Turing Machine. After scanning each line in the input scripts, the data structure is built up. In our system, the main elements in a Turning machine's data structure are listed below:

- **Input Tape.** A string denotes the read-only input tape.
- **Active Tapes.** A string array denotes the active tapes.
- **Initial State.** An integer denotes the index of initial state.
- **Final States.** An integer array denotes the final states.
- **Input alphabet.** A string array denotes input alphabets.
- **Internal States.** A string array denotes the internal states.
- **Active States.** An integer array denotes the active states.
- **Active Pointers.** An integer array of active tapes pointers.
- **Transition Table.** A string matrix of transition functions.

B. Interpreter

We have mainly implemented two types of operations when interpreting the input scripts. The first operation is "Next Word" operator while another one is "Get Words" operator. By definition, "Next Word" operator is defined through a function $NextWord(s, i, c)$ which takes the input script s, begin index i, and an end symbol c as parameters; It returns a substring s' that begins at index i and end with the character before end Symbol c. For example, let input script s be "type=TM;", $i = 0$ and c is '='. Then $NextWord(s, 0, '=')$ returns "type". At the same time, the begin index i will increase as $i' = i + |s'| + 1$. So now $i = 0 + 4 + 1 = 5$, we call $NextWord(s, 5, ';')$ will return "TM". Similarly, function $GetWords(s, i)$ which takes the input script s and begin index i as parameters; It returns a list of substrings $S = \{s_1, s_2, ...\}$ that begins at index i and each substring $s_k \in S$ is separated by ',' or ended by ';' in s. For example, let input script s be "state=00,01,11,10;" and $i = 6$, then $GetWords(s, i)$ will return $S = \{$"00", "01", "11", "10"$\}$.

Every time when the keyword of "transit=" is scanned in the input scripts, then the interpreter will repeat scanning all of the transition identifiers until the "}" is met. As shown in Figure 3, all transitions between every two status are stored into a string matrix $M_{N\times N}$ where N is number of the internal states. In this case, the Turing machine is defined by the scripts shown in Figure 2. So, there are $N = 4$ internal states namely q0, q1, q2, and q3. Every transition in the table is represented as a string $M_{i,j}$ in format of "Read/Write, Move" when there is one or more transitions from the i^{th} state to the j^{th} state. For example, according to the script there is one transition between q1 and a2 is "S(q1,X)=q2/1,->;". Therefore, $M_{1,2}$="X/1,->;"

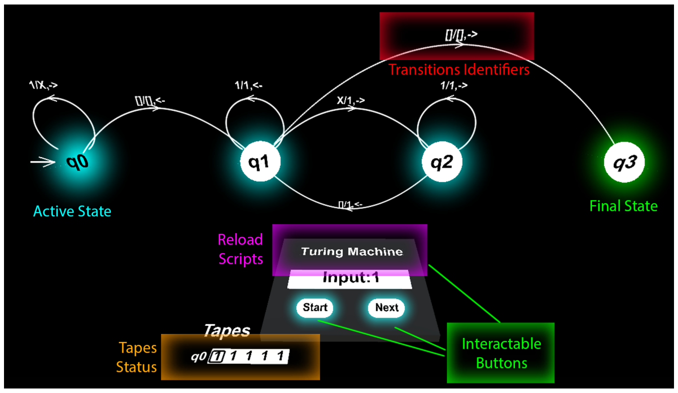


Fig. 4. Visualizations of our Turing machine simulator.

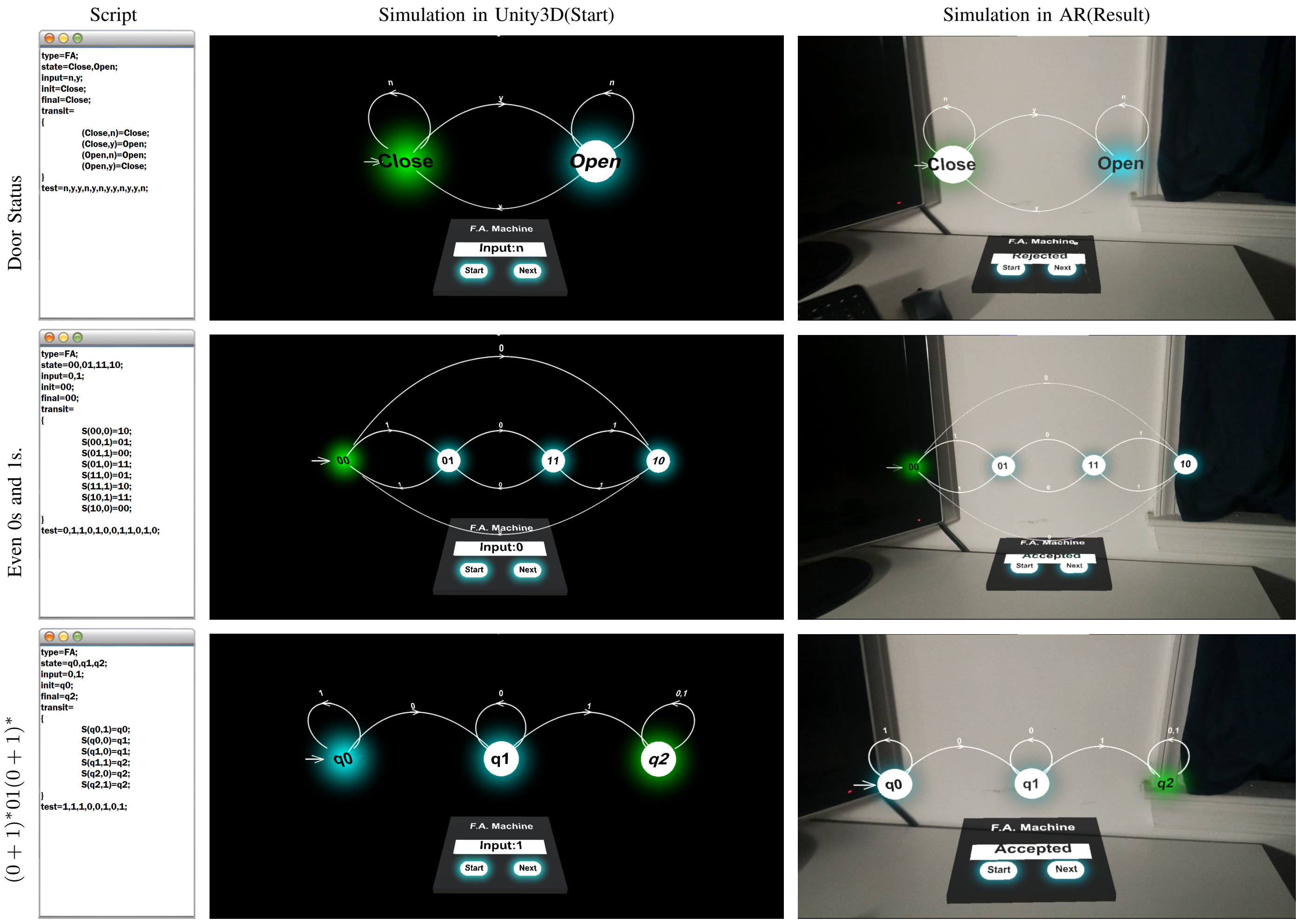
C. Visualizations

In order to visualize the data structure of the Turing machine, as considering the most commonly used diagram representation, those states are denoted as circles, transitions are denoted as curved arrows, states names, and transition conditions are labeled nearby the circles and curves. Specifically, our system proposes an automatic visualization approach that achieves a one-cut-fitting-all effect. That is: putting all different states in one horizontal line and connecting every two transition states with a parabolic trajectory. Such methods avoid heavy intersections among transition arrows automatically. Hereby, the key observations are surrounded by how to draw those parabolic trajectories smartly to avoid heavy intersections. One trick used here is to adjust the curvature of the parabolic trajectory according to how long two connected states are distant away from each other. The curvature of the parabolic trajectory increases as the distance increases and vice versa. Mathematically, the parabolic trajectory connecting the i^{th} state and the j^{th} state where there is a transition function defined from i to j is formulated as:

$$f^{T}_{i,j}(t) = (x_0 \pm k_x \frac{d_{i,j}t}{2T}, y_0 + k_y d_{i,j}(1 - \frac{t^2}{T^2})$$

where $d_{i,j}$ represents the distance between the centers of circles of the i^{th} state and the j^{th} state, (x_0, y_0) is the center position between i^{th} state and the j^{th} state, t is an arbitrary time when navigating along the trajectory and T is the total amount of time when navigating along the half trajectory (as it is symmetric). (k_x, k_y) is a vector to scale trajectory's range.

As shown in Figure 4, this is the visualization of Turing Machine using our method which depicts the example of Turing Machine that copies 1s on the tape. It is represented in our C language-like script as specified in Fig 2. The cyan glowing sphere represents the current active state is q0. The green circle denotes the q3 as the final state. Black squares on the tape denote the pointer position of the active tape. Note that there is one active tape corresponding to one active state. The active tapes are visualized on the white blocks (in the orange box), the transition identifiers (in the red box) are labeled above each transition curve.



```
type=FA;
state=Close,Open;
input=n,y;
init=Close;
final=Close;
transit=
{
            (Close,n)=Close;
            (Close,y)=Open;
            (Open,n)=Open;
            (Open,y)=Close;
}
test=n,y,y,n,y,n,y,y,n,y,y,n;
```

```
type=FA;
state=00,01,11,10;
input=0,1;
init=00;
final=00;
transit=
{
            S(00,0)=10;
            S(00,1)=01;
            S(01,1)=00;
            S(01,0)=11;
            S(11,0)=01;
            S(11,1)=10;
            S(10,1)=11;
            S(10,0)=00;
}
test=0,1,1,0,1,0,0,1,1,0,1,0;
```

```
type=FA;
state=q0,q1,q2;
input=0,1;
init=q0;
final=q2;
transit=
{
            S(q0,1)=q0;
            S(q0,0)=q1;
            S(q1,0)=q1;
            S(q1,1)=q2;
            S(q2,0)=q2;
            S(q2,1)=q2;
}
test=1,1,1,0,0,1,0,1;
```

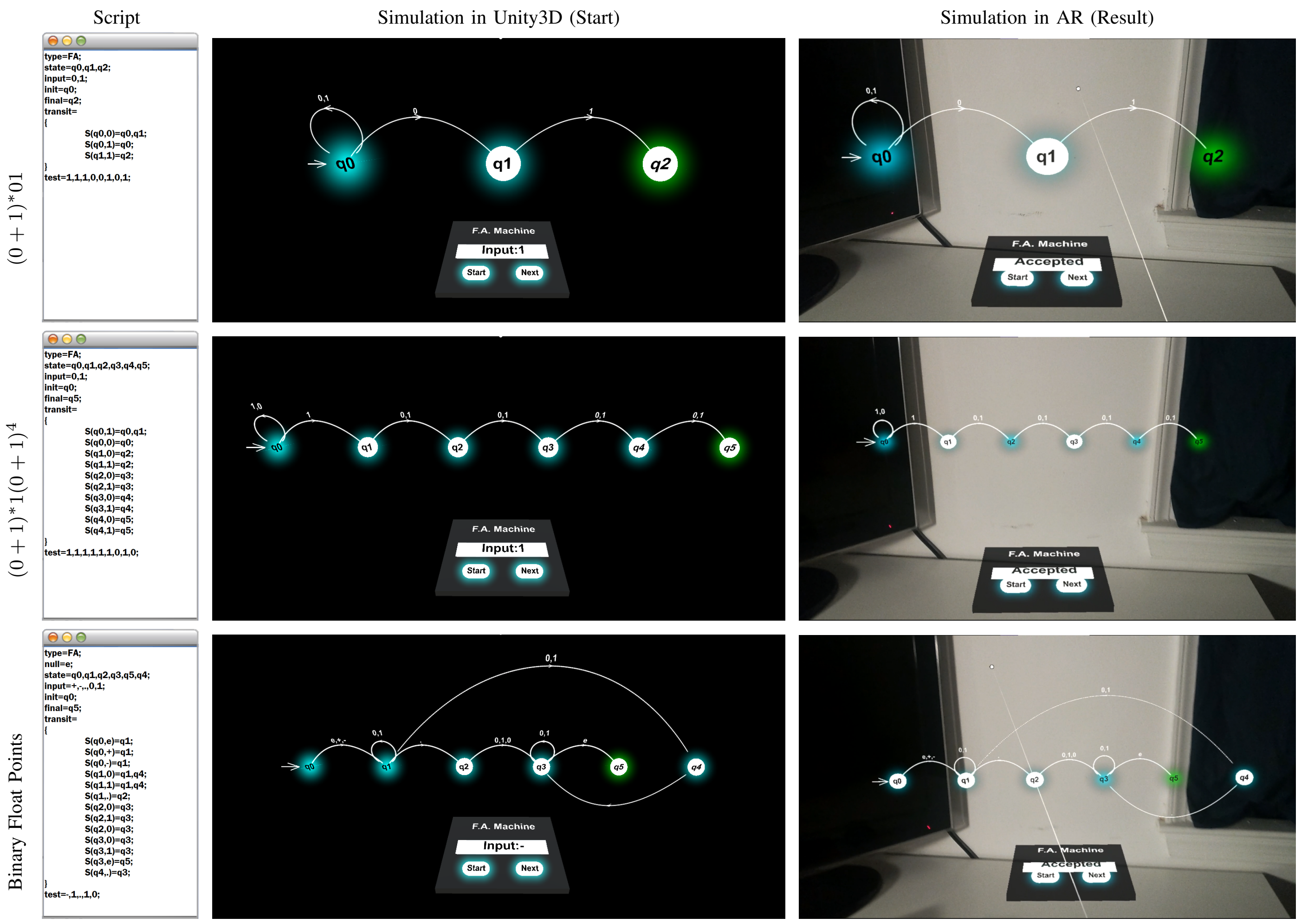
Fig. 5. Testing our proposed AR simulator with different types of Deterministic Finite Automaton (DFA).

D. Interactions

As shown in Figure 4, in order to design and simulate the Turing Machine effectively, we have developed several main interactive functions including (1) Reload scripts (in the purple box): when the label of ”F.A. Machine” (Finite Automaton), ”P.D.A. Machine” (Pushdown Automaton), or ”Turing Machine” displayed on the simulator is clicked, the C-like scripts in a specified text editor will be reloaded and a new simulator will be constructed. (2)Initialize the state: when the ”Start” button is clicked, the input string or tape will be reloaded and the simulator will be reset to the initial state. (3) Transit to the next states: when the ”Next” button is clicked, the current states on the simulator will be transited to the next status according to the transition functions specified in the C-like input scripts. During the simulation process, the HoloLens2 headset is connected with the desktop through a C-type USB cable, at the same time, the user can edit the C-like scripts through any text editor. After each update, the user may reload the script and reconstruct the simulator. Users wearing HoloLens2 can interact with the simulator with finger touch in the mid-air by clicking the mentioned interactable buttons.

V. Results

We demonstrate our results through an Augmented Reality (AR) device. As we know, AR is an interactive experience of a real-world environment where the objects that reside in the real world are enhanced by computer-generated perceptual information, sometimes across multiple sensory modalities, including visual, auditory, haptic, somatosensory, and olfactory [21]. Hereby, according to the nature of AR which is fun, immersive, and sometimes even addictive, we deploy our program onto an AR platform to enhance the education effects [22]. Here we take the advantages of HoloLens2 [23], a popular AR headset device developed by Microsoft, and developed the application through Mixed Reality Toolkits (MRTK) [24], which is a widely used AR plug-in on the Unity 3D game engine. We developed this program with one of the most popular development tools for eXtended Reality (XR) applications which are called Unity 3D [25]. As Unity 3D has integrated APIs which XR developers can easily have access to after installing the corresponding SDKs, the AR application can be efficiently developed with a library called Mixed Reality Toolkits (MRTK) which is a comprehensive,



```
type=FA;
state=q0,q1,q2;
input=0,1;
init=q0;
final=q2;
transit=
{
        S(q0,0)=q0,q1;
        S(q0,1)=q0;
        S(q1,1)=q2;
}
test=1,1,1,0,0,1,0,1;
```

```
type=FA;
state=q0,q1,q2,q3,q4,q5;
input=0,1;
init=q0;
final=q5;
transit=
{
        S(q0,1)=q0,q1;
        S(q0,0)=q0;
        S(q1,0)=q2;
        S(q1,1)=q2;
        S(q2,0)=q3;
        S(q2,1)=q3;
        S(q3,0)=q4;
        S(q3,1)=q4;
        S(q4,0)=q5;
        S(q4,1)=q5;
}
test=1,1,1,1,1,1,0,1,0;
```

```
type=FA;
null=e;
state=q0,q1,q2,q3,q5,q4;
input=+,-,.,0,1;
init=q0;
final=q5;
transit=
{
        S(q0,e)=q1;
        S(q0,+)=q1;
        S(q0,-)=q1;
        S(q1,0)=q1,q4;
        S(q1,1)=q1,q4;
        S(q1,.)=q2;
        S(q2,0)=q3;
        S(q2,1)=q3;
        S(q2,0)=q3;
        S(q3,0)=q3;
        S(q3,1)=q3;
        S(q3,e)=q5;
        S(q4,.)=q3;
}
test=-,1,.,1,0;
```

Fig. 6. Testing our proposed AR simulator with different types of Nondeterministic Finite Automaton (NFA).

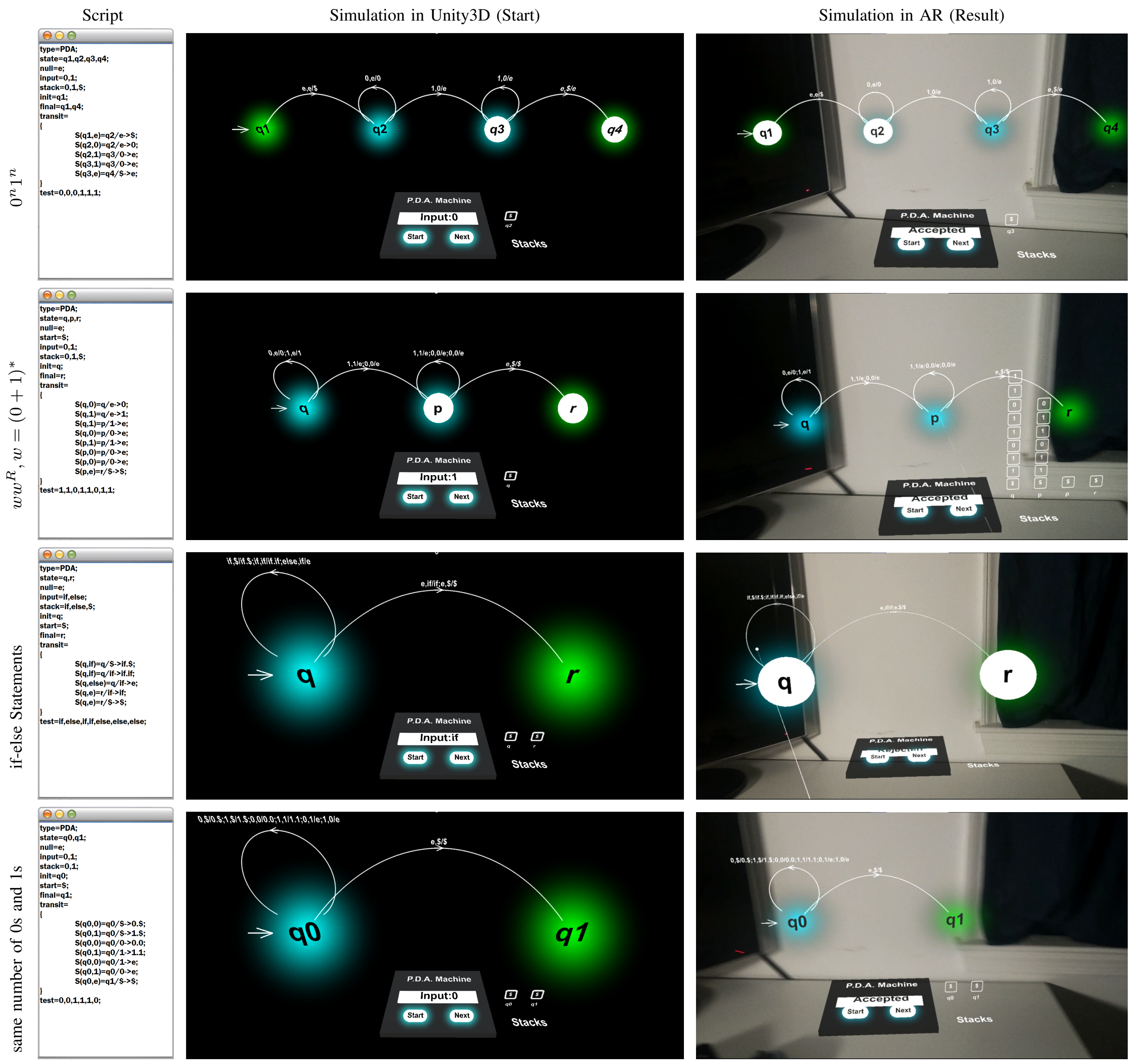
scalable enterprise AR platform [26]. MRTK's wide-ranging solution suites provide practical AR technology that is widely used among researchers nowadays.

In order to test the capabilities of our developed program, we have designed different types of automata and Turing Machine with our proposed C-like scripts and simulating them with different input strings or tapes. As shown in Figure 5, the left column shows the C-like script, the middle column shows the initialization of the simulator (Run in Unity 3D) and the right column shows the simulation result of the simulator given to the input string specified in the C-like script (Run on HoloLens2). Different rows demonstrate different types of automata or Turing machines respectively. Our system is designed for simulating different types of automata and Turing machines including Deterministic Finite Automata (DFA), Nondeterministic Finite Automata (NFA), Pushdown Automata (PDA), and Turing Machines (TM).

DFA. As shown in Figure 5, among the DFAs, the first one simulates the door status which is open or closed decided by whether there is any input (yes or no) to change the current state. The second DFA simulates the machine that accepts all 0-1 strings that contain an even number of 0s and an even number of 1s. The third DFA simulates the machine that accepts all 0-1 strings that contain a "01" substring, or, represented by a regular expression: $(0+1)^*01(0+1)^*$.

NFA. As shown in Figure 6, among the NFAs, the first one simulates the machine that accepts all 0-1 strings that end with a "01" substring, or, represented by a regular expression: $(0+1)^*01$. The second NFA simulates the machine that accepts all 0-1 strings whose 5th symbol from the right end is 1, or, represented by a regular expression: $(0+1)^*1(0+1)^4$. The third NFA simulates the machine that accepts all 0-1 strings representing binary floating-point numbers, e.g., -1.10.

PDA. As shown in Figure 7, among the PDAs, the first one simulates the machine that accepts all 0-1 strings that is represented by the regular expression: $0^n1^n, n = 1, 2, 3, ...$ The second PDA simulates the machine that accepts all 0-1 strings who are palindromes or, represented by a regular expression: $ww^R, w = (0+1)^*$. The third PDA simulates the machine that accepts all matching if-else statements. The last PDA simulates the machine that accepts 0-1 strings containing same number of 0s and 1s.



```
type=PDA;
state=q1,q2,q3,q4;
null=e;
input=0,1;
stack=0,1,$;
init=q1;
final=q1,q4;
transit=
{
        S(q1,e)=q2/e->$;
        S(q2,0)=q2/e->0;
        S(q2,1)=q3/0->e;
        S(q3,1)=q3/0->e;
        S(q3,e)=q4/$->e;
}
test=0,0,0,1,1,1;
```

```
type=PDA;
state=q,p,r;
null=e;
start=$;
input=0,1;
stack=0,1,$;
init=q;
final=r;
transit=
{
        S(q,0)=q/e->0;
        S(q,1)=q/e->1;
        S(q,1)=p/1->e;
        S(q,0)=p/0->e;
        S(p,1)=p/1->e;
        S(p,0)=p/0->e;
        S(p,0)=p/0->e;
        S(p,e)=r/$->$;
}
test=1,1,0,1,1,0,1,1;
```

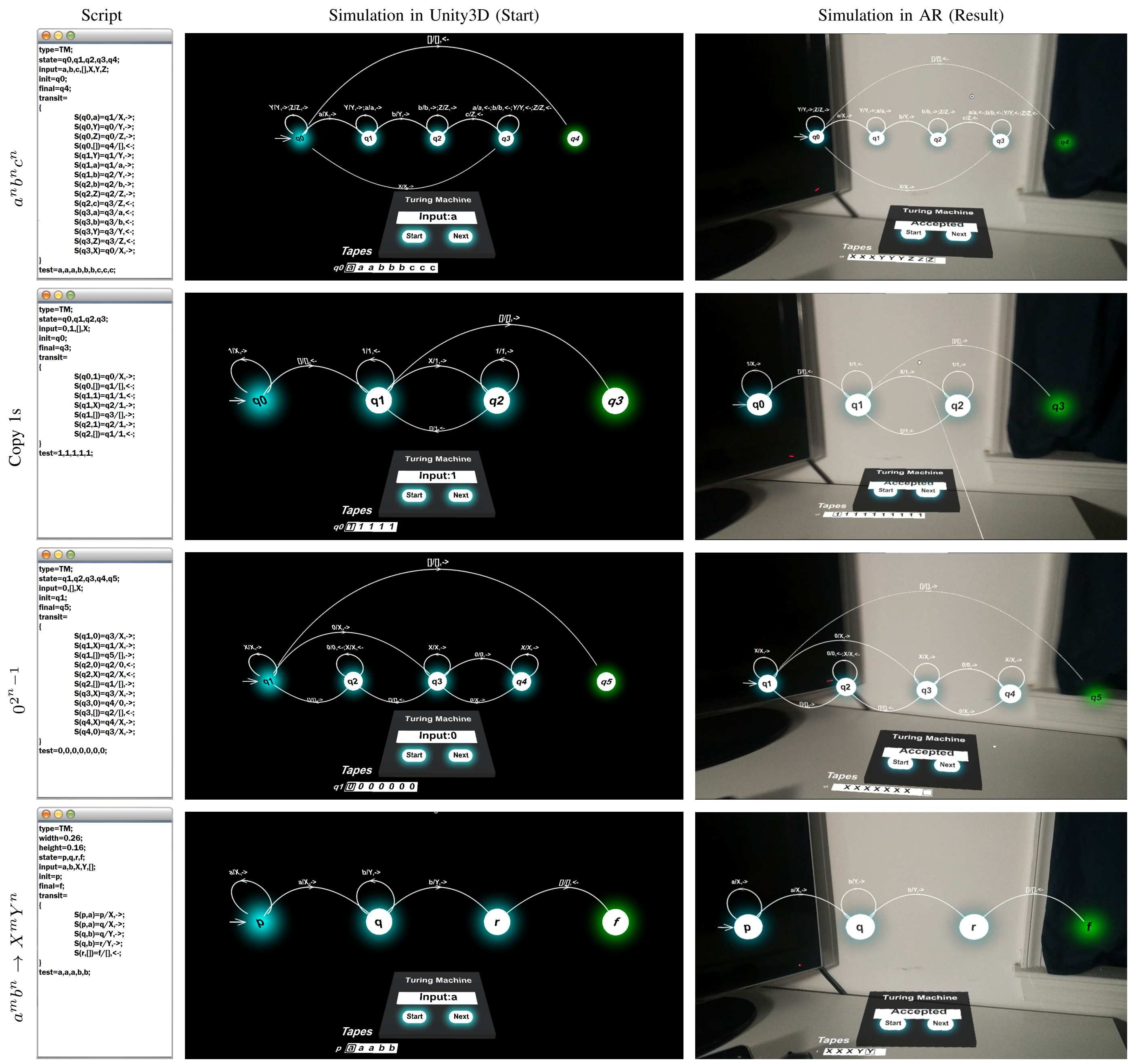
```
type=PDA;
state=q,r;
null=e;
input=if,else;
stack=if,else,$;
init=q;
start=$;
final=r;
transit=
{
        S(q,if)=q/$->if.$;
        S(q,if)=q/if->if.if;
        S(q,else)=q/if->e;
        S(q,e)=r/if->if;
        S(q,e)=r/$->$;
}
test=if,else,if,if,else,else,else;
```

```
type=PDA;
state=q0,q1;
null=e;
input=0,1;
stack=0,1;
init=q0;
start=$;
final=q1;
transit=
{
        S(q0,0)=q0/$->0.$;
        S(q0,1)=q0/$->1.$;
        S(q0,0)=q0/0->0.0;
        S(q0,1)=q0/1->1.1;
        S(q0,0)=q0/1->e;
        S(q0,1)=q0/0->e;
        S(q0,e)=q1/$->$;
}
test=0,0,1,1,1,0;
```

Fig. 7. Testing our proposed AR simulator with different types of Pushdown Automaton (PDA).

TM. As shown in Figure 8, among the TMs, the first one simulates the machine that accepts all a-b-c strings that are represented by the regular expression: $a^nb^nc^n, n = 1, 2, 3, \ldots$. The second TM simulates the machine that copies 1s. The third TM simulates the machine that accepts a string of 0s whose lengths are $2^n - 1, n = 1, 2, 3, \ldots$. The last TM is nondeterministic and simulates the machine that accepts $a^mb^n, m, n = 1, 2, 3, \ldots$ and replace $a \rightarrow X$ and $b \rightarrow Y$.

Comparison with related works. We compared our AR Turing machine simulator interface with some other existing Turing machine simulator interfaces. As shown in Figure 9, (a) demonstrates a command line-based TM simulator developed by Gourlay et al. [19]. In this interface, states are put in different lines on the console, different inputs are separated into different columns, tapes are printed at the top, and the active states are highlighted as white. However, this method is not straightforward as the Turing machine diagram is not visually presented. (b) shows a web page-based TM simulator developed by Wolfram et al. [20]. Despite the visualization is fancy enough to visualize the complex structure of a TM, however, it is over complicated for beginners to learn. Also, the way it presents the TM is quite different from the popular textbook such as the one written by Sipser et al. [2] which is widely welcomed by computer science students. As



```
type=TM;
state=q0,q1,q2,q3,q4;
input=a,b,c,[],X,Y,Z;
init=q0;
final=q4;
transit=
{
            S(q0,a)=q1/X,->;
            S(q0,Y)=q0/Y,->;
            S(q0,Z)=q0/Z,->;
            S(q0,[])=q4/[],<-;
            S(q1,Y)=q1/Y,->;
            S(q1,a)=q1/a,->;
            S(q1,b)=q2/Y,->;
            S(q2,b)=q2/b,->;
            S(q2,Z)=q2/Z,->;
            S(q2,c)=q3/Z,<-;
            S(q3,a)=q3/a,<-;
            S(q3,b)=q3/b,<-;
            S(q3,Y)=q3/Y,<-;
            S(q3,Z)=q3/Z,<-;
            S(q3,X)=q0/X,->;
}
test=a,a,a,b,b,b,c,c,c;
```

```
type=TM;
state=q0,q1,q2,q3;
input=0,1,[],X;
init=q0;
final=q3;
transit=
{
            S(q0,1)=q0/X,->;
            S(q0,[])=q1/[],<-;
            S(q1,1)=q1/1,<-;
            S(q1,X)=q2/1,->;
            S(q1,[])=q3/[],->;
            S(q2,1)=q2/1,->;
            S(q2,[])=q1/1,<-;
}
test=1,1,1,1,1;
```

```
type=TM;
state=q1,q2,q3,q4,q5;
input=0,[],X;
init=q1;
final=q5;
transit=
{
            S(q1,0)=q3/X,->;
            S(q1,X)=q1/X,->;
            S(q1,[])=q5/[],->;
            S(q2,0)=q2/0,<-;
            S(q2,X)=q2/X,<-;
            S(q2,[])=q1/[],->;
            S(q3,X)=q3/X,->;
            S(q3,0)=q4/0,->;
            S(q3,[])=q2/[],<-;
            S(q4,X)=q4/X,->;
            S(q4,0)=q3/X,->;
}
test=0,0,0,0,0,0,0;
```

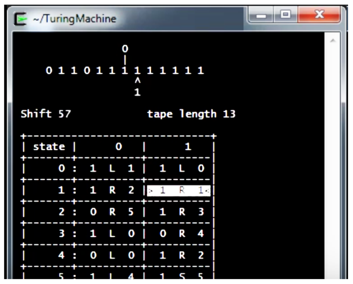
```
type=TM;
width=0.26;
height=0.16;
state=p,q,r,f;
input=a,b,X,Y,[];
init=p;
final=f;
transit=
{
            S(p,a)=p/X,->;
            S(p,a)=q/X,->;
            S(q,b)=q/Y,->;
            S(q,b)=r/Y,->;
            S(r,[])=f/[],<-;
}
test=a,a,a,b,b;
```

Fig. 8. Testing our proposed AR simulator with different types of Turing machines.

another web page-based TM simulator, (c) shows the web app developed by Morphett et al. [18]. As we can see, the script to design the TM is not straightforward and is hard to follow for most beginners as well. Compared to these works, our approach presents the Turing machine more immersively.

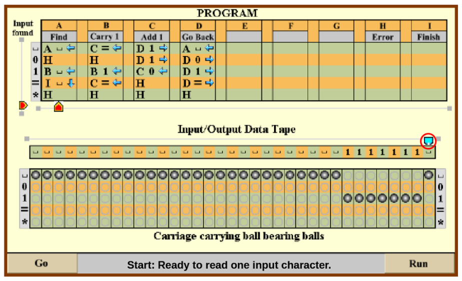
VI. Conclusion and Future Work

Through an immersive AR device called Microsoft HoloLens2 which is a pair of look-through glasses for mixed reality with a head-mounted display that can render virtual objects upon the real-world scene, we have successfully simulated the automaton and Turing machine in Augmented Reality. Users wearing HoloLens2 can interact with virtual automaton and Turing machine simulator and update the script in the real world simultaneously. In our proposed HoloLens-based AR interface for Turing machine simulation, users are able to observe and operate the transitions between different configurations or states of the virtual Turing machine, such simulations taking place on the table instead of on the computer screen will leave the user with impressive feelings. At the same time, augmented reality is unlike mere virtual reality as users are not able to see and operate any real objects in

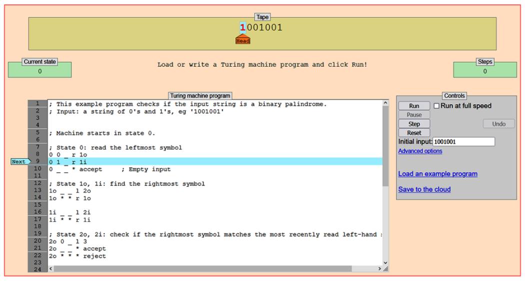


```
                0
                |
    0 1 1 0 1 1 1 1 1 1 1 1 1
                  ^
                  1

Shift 57              tape length 13

+-----------------------------+
| state |     0     |    1    |
+-------+-----------+---------|
|    0 :  1  L  1 |  1  L  0 |
+-------+-----------+---------|
|    1 :  1  R  2 |  1  R  1 |
+-------+-----------+---------|
|    2 :  0  R  5 |  1  R  3 |
+-------+-----------+---------|
|    3 :  1  L  0 |  0  R  4 |
+-------+-----------+---------|
|    4 :  0  L  0 |  1  R  2 |
+-------+-----------+---------|
|    5 :  1  L  4 |  1  S  5 |
```

(a) By Gourlay et al. [19]



(b) By Wolfram et al. [20]



Initial input: 1001001

```
; This example program checks if the input string is a binary palindrome.
; Input: a string of 0's and 1's, eg '1001001'


; Machine starts in state 0.

; State 0: read the leftmost symbol
0 0 _ r 1o
0 1 _ r 1i
0 _ _ * accept     ; Empty input

; State 1o, 1i: find the rightmost symbol
1o _ _ l 2o
1o * * r 1o

1i _ _ l 2i
1i * * r 1i

; State 2o, 2i: check if the rightmost symbol matches the most recently read left-hand :
2o 0 _ l 3
2o _ _ * accept
2o * * * reject
```

(c) By Morphett et al. [18]

Fig. 9. Some existing Turing machine simulator interfaces.

virtual reality, while augmented reality gives users the freedom to type on a keyboard, coding, and debugging through the computer screen just like wearing a normal transparent glasses. Our well-designed interactive interface provides a more realistic and user-friendly platform for students to learn and explore the fascinating mathematics world of automation and computational theories. We believe our work can be extended and well-studied in the following future works to bring the computational theory teaching industry into an entirely new AR age, such that every student enrolled in theory classes in the future will no more think the lecture is boring and abstract, rather, they will enjoy the new concepts taught through AR.

References

[1] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," *J. of Math*, vol. 58, no. 345-363, p. 5, 1936.

[2] M. Sipser, "Introduction to the theory of computation," *ACM Sigact News*, vol. 27, no. 1, pp. 27–29, 1996.

[3] J. Tan, B. Gao, and X. Lu, "An ar system for artistic creativity education," in *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, 2018, pp. 1–2.

[4] I. Kico and F. Liarokapis, "A mobile augmented reality interface for teaching folk dances," in *25th ACM Symposium on Virtual Reality Software and Technology*, 2019, pp. 1–2.

[5] C. Sinnott, J. Liu, C. Matera, S. Halow, A. Jones, M. Moroz, J. Mulligan, M. Crognale, E. Folmer, and P. MacNeilage, "Underwater virtual reality system for neutral buoyancy training: Development and evaluation," in *25th ACM Symposium on Virtual Reality Software and Technology*, 2019, pp. 1–9.

[6] L. Jensen and F. Konradsen, "A review of the use of virtual reality head-mounted displays in education and training," *Education and Information Technologies*, vol. 23, no. 4, pp. 1515–1529, 2018.

[7] A. Zable, L. Hollenberg, E. Velloso, and J. Goncalves, "Investigating immersive virtual reality as an educational tool for quantum computing," in *26th ACM Symposium on Virtual Reality Software and Technology*, 2020, pp. 1–11.

[8] F. C. Hennie and R. E. Stearns, "Two-tape simulation of multitape turing machines," *Journal of the ACM (JACM)*, vol. 13, no. 4, pp. 533–546, 1966.

[9] J. Robson, "Fast probabilistic ram simulation of single tape turing machine computations," *Information and control*, vol. 63, no. 1-2, pp. 67–87, 1984.

[12] M. Carpentieri, "On the simulation of quantum turing machines," *Theoretical computer science*, vol. 304, no. 1-3, pp. 103–128, 2003.

[10] M. Dauchet, "Simulation of turing machines by a left-linear rewrite rule," in *International Conference on Rewriting Techniques and Applications*. Springer, 1989, pp. 109–120.

[11] ——, "Simulation of turing machines by a regular rewrite rule," *Theoretical computer science*, vol. 103, no. 2, pp. 409–420, 1992.

[13] Y. Zhu, F. Pan, L. Li, X. Ren, and Q. Gao, "Simulation of turing machine with ueac-computable functions," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[14] A. Dengel, "Seeking the treasures of theoretical computer science education: Towards educational virtual reality for the visualization of finite state machines," in *2018 IEEE international conference on teaching, assessment, and learning for engineering (TALE)*. IEEE, 2018, pp. 1107–1112.

[15] J. Pirker, A. Dengel, M. Holly, and S. Safikhani, "Virtual reality in computer science education: A systematic review," in *26th ACM Symposium on Virtual Reality Software and Technology*, 2020, pp. 1–8.

[16] A. Wohlan, N. Hochgeschwender, and N. Meissler, "Visualizing convolutional neural networks with virtual reality," in *25th ACM Symposium on Virtual Reality Software and Technology*, 2019, pp. 1–2.

[17] M. Hamada, "An example of virtual environment and web-based application in learning." *IJVR*, vol. 7, no. 3, pp. 1–8, 2008.

[18] A. Morphett. Turing machine simulator. [Online]. Available: http://morphett.info/turing/turing.html

[19] M. J. Gourlay. Turing machine simulator. [Online]. Available: https://www.mijagourlay.com/turingmachine

[20] W. D. Project. Simulation of a turing machine. [Online]. Available: https://demonstrations.wolfram.com/SimulationOfATuringMachine/

[21] S. Zlatanova, "Augmented reality technology," *GISt Report No. 17, Delft, 2002, 72 p.*, 2002.

[22] M. Billinghurst, "Augmented reality in education," *New horizons for learning*, vol. 12, no. 5, pp. 1–5, 2002.

[23] A. Vidal-Balea, O. Blanco-Novoa, I. Picallo-Guembe, M. Celaya-Echarri, P. Fraga-Lamas, P. Lopez-Iturri, L. Azpilicueta, F. Falcone, and T. M. Fernández-Caramés, "Analysis, design and practical validation of an augmented reality teaching system based on microsoft hololens 2 and edge computing," in *Engineering Proceedings*, vol. 2, no. 1. Multidisciplinary Digital Publishing Institute, 2020, p. 52.

[24] K. Lotsaris, C. Gkournelos, N. Fousekis, N. Kousi, and S. Makris, "Ar based robot programming using teaching by demonstration techniques," *Procedia CIRP*, vol. 97, pp. 459–463, 2021.

[25] J. Linowes, *Unity virtual reality projects*. Packt Publishing Ltd, 2015.

[26] P. Trentsios, M. Wolf, and D. Gerhard, "Comparing virtual reality sdk potentials for engineering education," in *International Conference on Remote Engineering and Virtual Instrumentation*. Springer, 2020, pp. 375–392.