By Richard LaFranchi <rlafranc@rams.colostate.edu>

Systematic Test Case Generation For Augmented Reality Applications

---

Background

Augmented Reality (AR) applications using mobile phones and AR specific hardware such as the HoloLens have been making big advancements in what is possible for new types of software. AR is software that uses a camera and places virtual objects in a real environment. Testing augmented reality applications is challenging because of the reliance on physical hardware and the need of a real environment. This project will evaluate potential solutions for testing AR applications and will specifically evaluate techniques for systematic test case generation.

Problem Statement

Novel techniques exist for testing traditional 2D GUI applications, but very little research exists on testing Augmented Reality applications and more specifically automated testing. The problem that we are trying to solve is whether or not these traditional techniques will be effective on AR applications. Existing research for testing AR/VR applications exist such as "An open software architecture for virtual reality interaction" **[1] &** "An automated functional testing approach for virtual reality applications**" [2]. [1]** describes an open source tool for interacting with VR environments, but appears to be outdated, with broken links to the project, and not well documented. **[2]** describes a testing framework that presents a design for testing AR/VR software, but is very vague on describing actual implementation with no actual data on results. Although there exists research in these areas, breadth of these topics is lacking. Instead, a better approach that we want to take is to apply tried and true techniques for testing 2D GUIs.

Research Question (note- research question is required for CS567) - **Can existing techniques for testing 2D GUI applications be effective on testing Augmented Reality applications?**

Approaches

We will investigate existing techniques for GUI testing and apply them to an Augmented Reality environment. The hope is to discover what can be achieved using these techniques and what can't be achieved in hopes to lead the way to more research in this area. The approach that we will follow can be described in the paper "Automating Regression Testing for Evolving GUI Software" **[3]** which describes an end to end process for automated regression testing. A Part of the process includes a technique called GUI Ripping **[4]** which uses a depth-first traversal algorithm for traversing elements such as menu items, buttons, and other elements for

generating event-based test cases. In an AR environment, both 2D objects such as buttons can exist along with 3D objects. The goal is to see if the GUI Ripping can be used as a method for discovering 3D elements and use this to generate test cases as it relates to 3D objects. This project is being done in conjunction with CS567 and Professor Ortega. As a part of this, an AR application for measuring the exterior of buildings and structures will be developed as a case study for using the described technique. The approach will involve an algorithm for discovering 3D objects along with test generation. The process does not include automating test execution since automation is not trivial in AR environments.

Evaluation/Metrics

The challenge in using 2D GUI techniques on AR applications is that 2D GUIs are more or less static, that is the elements in the GUI such as buttons, labels, and inputs don't change. An AR application is very much dynamic, so 3D objects or elements may or may not be visible depending on the current state of the environment. This means that generation of the test cases will be dynamic as well. We may have to repeat a process multiple times and calculate an average number of test cases generated on a repeated task. For the AR application of measuring exterior walls of buildings, we can test against multiple buildings, and different starting locations in the environment. The main metric that will be evaluated will be number of test cases generated based on different scenarios. Test cases that are generated will be executed manually and from the execution we will also evaluate. Depending on the number of test cases generated, a subset may be chosen if manual execution is too time consuming. From the execution we will evaluate number of faults discovered.

Tasks/Tools

Apple's ARKit and XCode will be used as a development and environment. The development of that actual application is mostly complete, so the goal is to build in a test mode for test generation. Since a real environment is required, the idea is to manually walk around buildings, but include some code for generating the test cases without actually manually selecting 3D objects. 3 Main tasks can be described:

1. Develop GUI Ripper/Traversal algorithm for 3D Nodes/Objects in XCode (will be manually implemented as a test mode in an AR application)
2. Develop Reporting mechanism for test generation
3. Perform Experiments and manual test case execution on multiple buildings/scenarios

Timeline

Date	**Task/Description**

11/4	**Task 1 - basic algorithm for traversing 3D nodes/objects**
11/18	**Task 2 - develop way to report test cases and report them**
12/6	**Task 3 - complete experiments/collect results**
12/13	**Complete Video Presentation and Paper**

References

[1] Gerhard Reitmayr and Dieter Schmalstieg. 2001. An open software architecture for virtual reality interaction. In Proceedings of the ACM symposium on Virtual reality software and technology (VRST '01). ACM, New York, NY, USA, 47-54. DOI=http://dx.doi.org/10.1145/505008.505018

[2] Souza ACC, Nunes FLS, Delamaro ME. An automated functional testing approach for virtual reality applications. Softw Test Verif Reliab. 2018;28:e1690. https://doi.org/10.1002/stvr.1690

[3] Memon, A. , Nagarajan, A. and Xie, Q. (2005), Automating regression testing for evolving GUI software. J. Softw. Maint. Evol.: Res. Pract., 17: 27-64. doi:10.1002/smr.305

[4] GUI ripping: Reverse engineering of graphical user interfaces for testing (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.122.5392&rep=rep1&type=pdf)