IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received November 8, 2017, accepted December 30, 2017, date of publication January 16, 2018, date of current version March 9, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2794301

Robotic Arm Based 3D Reconstruction Test Automation

DEBDEEP BANERJEE, KEVIN YU, AND GARIMA AGGARWAL
Qualcomm Technologies Inc., San Diego, CA 92121, USA

Corresponding author: Debdeep Banerjee (debdeepb@qti.qualcomm.com)

ABSTRACT The 3-D reconstruction involves the construction of a 3-D model from a set of images. The 3-D reconstruction has varied uses that include 3-D printing, the generation of 3-D models that can be shared through social media, and more. The 3-D reconstruction involves complex computations in mobile phones that must determine the pose estimation. The pose estimation involves the process of transforming a 2-D object into 3-D space. Once the pose estimation is done, then the mesh generation is performed using the graphics processing unit. This helps render the 3-D object. The competitive advantages of using hardware processors are to accelerate the intensive computation using graphics processors and digital signal processors. The stated problem that this technical paper addresses is the need for a reliable automated test for the 3-D reconstruction feature. The solution to this problem involved the design and development of an automated test system using a programmable robotic arm and rotor for precisely testing the quality of 3-D reconstruction features. The 3-D reconstruction testing involves using a robotic arm lab to accurately test the algorithmic integrity and end-to-end validation of the generated 3-D models. The robotic arm can move the hardware at different panning speeds, specific angles, fixed distances from the object, and more. The ability to reproduce the scanning at a fixed distance and the same panning speed helps to generate test results that can be benchmarked by different software builds. The 3-D reconstruction also requires a depth sensor to be mounted onto the device under examination. We use this robotic arm lab for functional, high performance, and stable validation of the 3-D reconstruction feature. This paper addresses the computer vision use case testing for 3-D reconstruction features and how we have used the robotic arm lab for automating these use cases.

INDEX TERMS Software engineering, software testing, computer vision, robotics and automation, robots.

I. INTRODUCTION

This technical paper focuses on automated testing computer vision features such as 3D reconstruction. We effectively use the robotic arm lab to facilitate the test automation.

In computer vision, 3D reconstruction is the process of capturing the shape and appearance of real objects. This process can be accomplished either by active or passive methods. If the model is allowed to change its shape in time, then it is referred to as non-rigid or spatiotemporal reconstruction.

The 3D reconstruction algorithms implemented in the Qualcomm Technologies Inc.'s hardware deliverables include the processing in the CPU/GPU (Graphics Processing Unit). Some of the solutions offered include RGBD (D stands for depth information) tracking, flexible region of interest selection, and more. The figure for 3D reconstruction (Figure 1) shows the steps for digitally generating a 3D model; 3D printing and online sharing are some of the use cases for this feature.

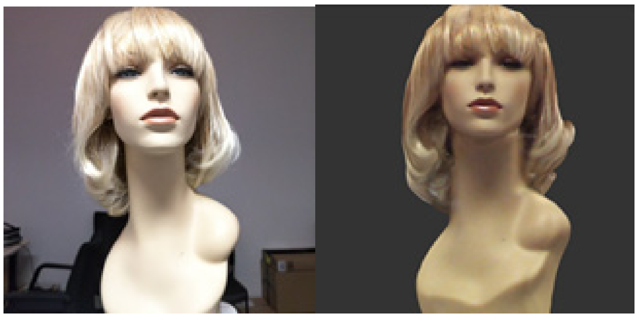

FIGURE 1. Depicts the different phases of 3D reconstruction. It includes the steps of mesh generation and color correction.

3D reconstruction test automation involves a comprehensive software deployment that executes the test cases; they detect the location of the device under test and then use the robotic arm to pick up the hardware. It then reads the test configuration and launches the 3D reconstruction application. The robotic arm starts panning the object's hardware. The robotic arm has 6 joints and it has been programmed to pan the hardware at specific panning speeds, different angles and different distances.

The 3D reconstruction feature is tested at two levels. The first level uses the API level test application that validates the algorithmic integrity of the computer vision and graphics functions used in the 3D reconstruction application. The other level of tests involves the validation of the 3D reconstruction feature using an end-to-end application of the camera sensor input.

The automated test may be deployed in various phases of the software integration so that any new regressions can be caught early in the software build and integrated into the test's life cycle. The goal is to insert automated tests in continuous integration tests so that any code submitted by the software developers is tested. Specific 3D reconstruction tests are executed on these new code changes and any regressions can be debugged and fixed. This leads to a lower time to catch and fix new regression issues.

II. MOTIVATION

The testing of 3D reconstruction features at various gates of the software development lifecycle is critical for successful commercialization of this feature. The motivation for the development of the 3D reconstruction test automation was due to the following points:

A. VALIDATION WITH PANNING OF THE DEVICE WITH A FIXED SPEED

The panning speed must be the same to conduct performance tests. It is difficult to maintain the same panning speed while manually performing the test; therefore, the robotic arm provides us with an excellent opportunity to accurately perform these performance tests. With the same panning speed and fixed distance from the test object, we should accept that the software engine will generate the same 3D model for every test.

B. VALIDATION WITH FIXED DISTANCE FROM OBJECT

The panning of the test objects must be a fixed distance from the device when conducting performance testing. If there are variations in the distance, the 3D model generated may have deformations or errors. Additionally, we can calculate pose estimation at specific distances of the object from the device. This can be easily achieved by using the robotic arm setup.

C. VALIDATION WITH SAME ORIENTATION OF THE TEST OBJECT

The object under test may not move and the panning angle between the phone and the object should be kept constant when conducting performance tests.

The goal of these performance tests is to be able to replicate the same results under the same test conditions, such as panning angle, distance, and speed. In this way we, should be able to generate the same 3D model every time during these tests and benchmark the test results across multiple builds.

D. VALIDATION WITH DIFFERENT CAMERA 3D SENSORS

The challenge in testing 3D reconstruction is that we need to employ multiple camera 3D sensors. We have plans to test 3D sensors that can be connected to a USB port and easily mounted on phones. We can program the location of the holding the device under test using the robotic arm. We can then ensure that the 3D depth sensor is connected to the USB port using a cable. We program the robotic arm in such a way that it does not hold the device where the USB socket is present. In this way, the mounting of the depth sensor is facilitated.

III. BACKGROUND AND RELATED WORK

The background to this technical paper reinforces the need for reliable and accurate test automation for 3D reconstruction features. We will discuss the presence of the gap between what is expected as an end-to-end automated test system for 3D reconstruction testing and manual testing of 3D reconstruction features. We have observed that manual testing for 3D reconstruction is not accurate since it is difficult to precisely maintain a fixed distance of the 3D scanner (using the stereo camera of the phone) from the tested 3D object, a fixed panning speed and fixed panning angles. The automated test for 3D reconstruction involves stereo camera based depth sensors for accurately calculating the distance of the camera phone from the 3D object for each frame. With the explosion of 3D based model generation and 3D scanning, it is of the utmost importance to design and develop reliable test automation systems to accurately test these 3D model generations and quantify their quality.

There have been tests conducted on 3D TV monitors. The discussions include the conditions that should be applied for standard stereo visual acuity tests. These tests can be executed on 3D TV monitors. The role of environmental lighting conditions on the measurement of stereo visual acuity when using conventional Stereoscopic 3D tests has been emphasized [1].

The effect of luminance and contrast on monocular or binocular visual acuity [2]–[4] is studied. In parallel, the recent revival of interest for stereoscopic vision and the related visual discomfort has been discussed [5], [6]. It has focused the attention on stereoscopic vision, justifying a fine analysis of 3D perception dysfunctions, defining the appropriate tests and protocols to measure them and providing recommendations [7]. For testing these features on mobile phones, robots have been used. It can involve a physical robot for testing overall smartphone device performance [8]. Profiling deployed SW for testing was discussed extensively in [9].

Several efforts [10]–[13] have applied symbolic execution [14]–[16] to generate inputs to Android apps.

Model-based testing has been widely studied in testing GUI-based programs [17]–[20]. In general, model-based testing requires users to provide a model of the app's

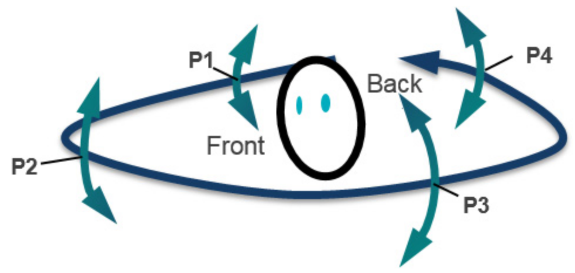

FIGURE 2. Guided path for scanning a person's head.

GUI [21], [22] though automated GUI model inference tools tailored to specific GUI frameworks also exist [23], [24].

Software testing based on GUI events facilitate the end-to-end automation of applications. Robot-based software testing is becoming prevalent since it facilitates accurate and precise testing of repetitive tasks.

IV. SOLUTION

A. OVERVIEW

3D reconstruction is an object scanning feature that allows users to scan a human head or any object in your room and generate a 3D image model of the object. The output meshes are saved in OBJ format. The key differentiators of this feature are:

- It allows RGBD tracking (which includes depth information with RGB data).
- It has online feedback for the reconstructed model.
- It also provides high definition textural mapping.

There are a variety of use cases for 3D reconstruction. The most significant ones are creating an image model for 3D-printing and sharing the model through social network.

B. OBJECTIVE

From a testing perspective, our main goal is to test the 3D reconstruction feature's functionality, quality, applicability, and stability. It should give us a clear high definition image model. It is important for the feature to be easy to use, which should not take more than two rounds to complete the scan. Here are the guidelines for manually scanning the object using a device with depth sensor:

- Scan an area where it has top diffused light. An average of 300 diffused LUX should be enough. Avoid direct lighting and sunlight.
- Move slowly around the test subject and scan the side and top according to the guided path (Fig. 2).
- If scanning an object on a surface, make sure that the surface is not reflective.

According to the recommended guidelines for the 3D reconstruction scanning, the depth camera mounted on the device should scan around the test subject with a proper distance of between 50 cm to 80 cm. The scanning should cover the top and bottom of the test subject for a complete 3D model. For example, in the case of scanning a human head, we need to scan 360 degrees around the head and stop at every 90 degrees at P1, P2, P3 and P4 to scan the top and bottom part of the head. After scanning through the guided path, the software has gathered enough depth and color information from the test subject to generate a 3D model.

This process may not be too complicated for manual functional verification testing. However, for the stability test scenario, we want to catch any bugs that can cause the app to freeze or crash. This is accomplished by repeating this scanning process for up to 300 iterations. Then, testing by manual scanning is too much work and too time-consuming for the testers. For the quality profiling scenario, we want to obtain perfect scans of the test subject for comparing different solutions and different sensors on the market. This requires that the 3D scan follows the exact same scanning path each time, which is also impossible when manually holding the device and walking around the test subject while scanning. Therefore, we created an automated testing system that uses a robotic arm to do the scanning. This requires less testing space, produces more consistent scans and requires significantly less manual interaction.

C. 3D RECONSTRUCTION TEST AUTOMATION SETUP

In our 3D reconstruction test automation setup, there are two key components: the rotor and the robotic arm. The rotor's purpose is to control the rotation of the test platform. Since any depth sensor on the market will require at least 50 cm scanning distance from the object for quality depth information, scanning by circling the object is physically impossible for most of the robotic arms. We have tried to hang the object on center top of the robotic arm and let it circle scan from the outer loop. However, it is still difficult to scan from a distance greater from the object than the minimum and retain the same level as the object. It is certainly not feasible to scan the top of the object for any small and midsize robot arm. Therefore, we thought of an innovative way to scan the object using a robotic arm. Instead of scanning around the object, we keep the depth sensor stationary and let the object rotate by placing the object on a programmable rotor. This way the sensor will be able to obtain a scan around the object. The biggest advantage of this type of a setup is it offers the freedom to adjust the scanning distances without physical limitations. Furthermore, the scanning distance is consistent during the scanning process (Please refer to Fig. 3).

The other key component of the test setup is the robotic arm. It holds the device with the depth sensor at the proper level of the object for the round scan. The level of the device is adjustable by programming the robotic arm for small, medium and large objects. However, the main purpose of the robotic arm is to lift the device with the depth sensor above the scanning objects to scan the top part of them. By combining the rotor and the robotic arm, we can get a complete 3D scan of an object with the suggested guided path.

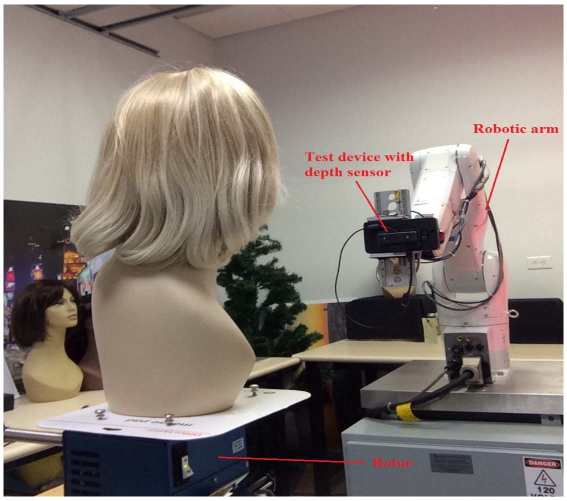

FIGURE 3. 3D scan automation system setup.

D. TEST AUTOMATION ALGORITHM

Our current test automation enables most of the testing procedure. Before starting the automation, the chosen test object needs to be placed on the center of the rotor platform for scanning. Then, the test device with depth sensor needs to establish a bridge connection over Wi-Fi or a USB cable to communication with the test station. The test station is also connected with the rotor setup and the robotic arm to control them using an automation script.

First, the automation script sends a command to the robotic arm controller. The execution of the command makes the robotic arm pick up the device under test. According to the test parameters, the script triggers the corresponding pre-generated robot programs to move the test device to the scanning position facing the test object with the same approximate level. After the device position is ready, the test script sends commands to the test device to start the scanning at P1. Then, the script triggers the rotor, and set it to rotate 90 degrees at the desired speed. The rotor stops for a certain amount of time at P2, waits for the robotic arm to lift the device to the top of the test object and down to scan the top of the object. Then, it rotates another 90 degrees to P3. It repeats this process for P3 and P4 until it reaches P1 again. At this point, the automation completes a 360 degree rotation and the top scans at P1, P2, P3 and P4. The script enables the device to generate a 3D model of the test object. The script also collects relevant logs of the 3D scan and then saves them.

The automation program will loop through the pre-set number of iterations and repeat the scanning process to reach the iteration count. It will return the device back to where it picked it up after the iterations are complete; the arm then moves back to original resting position. The automation script will collect the test result files and upload them to the defined result location for the testers to evaluate. Fig. 4 outlines the program's algorithm.

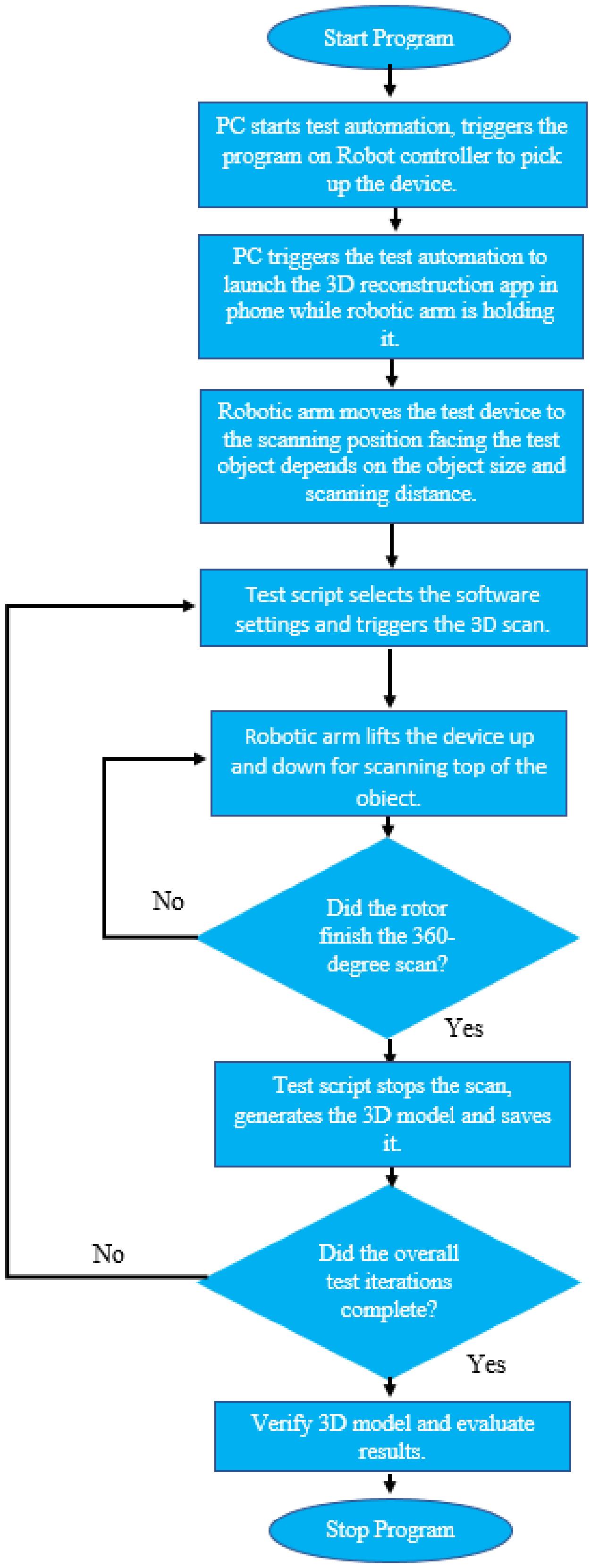
Start Program
PC starts test automation, triggers the program on Robot controller to pick up the device.
PC triggers the test automation to launch the 3D reconstruction app in phone while robotic arm is holding it.
Robotic arm moves the test device to the scanning position facing the test object depends on the object size and scanning distance.
Test script selects the software settings and triggers the 3D scan.
Robotic arm lifts the device up and down for scanning top of the object.
Did the rotor finish the 360-degree scan?
No
Yes
Test script stops the scan, generates the 3D model and saves it.
Did the overall test iterations complete?
No
Yes
Verify 3D model and evaluate results.
Stop Program

FIGURE 4. Robotic Arm test automation program algorithm.

E. PROGRAMMING ROBOTIC ARM POSITIONS

Here are some insights on how the Denso robotic arm was programmed to perform the scanning function. We program the robotic arm using the WINCAPS 3 software and its built-in option for the 3D arm view. Denso Robotics developed the WINCAPS3 software. We can simulate the robotic arm's concurrent positions and movements with coordinates using the WINCAPS3 software. We find it to be very helpful to create 3D models in the simulated environment. In 3D reconstruction testing, different test objects and various scanning distances are always needed. These 3D models' sizes and coordinates are a good representation of the actual test setup. They can be used for relative positions when programming the robotic arm.

In our test plan, we have three categories of test object sizes: small, medium and large. Additionally, our scanning distance ranges from 40 cm to 80 cm. A robot program is needed for each of these categories and distances. In the case of the large sized model with a scanning distance of 60 cm, we measure the height of the rotor to be 25 cm and the model height as 50 cm. The rotor is represented by the cube with its relative size, and the model is represented by the cylinder and sphere with their relative sizes. The rotor setup is placed 100 cm away from the base of the robotic arm. We need to calculate how far to extend the tip of the robotic arm so that the sensor is placed 60 cm away from the model and the sensor's altitude that faces parallel to the model for a good scan coverage, as indicated in Fig. 5a.

From the diagram in Fig. 5a, we can see that the rotor with the test model is being set up aligned with the base of the robotic arm for simplification. The position of the tip of the robotic arm should have a Y-axis value close to 0. The distance that the robotic arm should extend out should be 100 cm – 60 cm = 40 cm. Since the robot is facing backwards, the X-axis value should be −40 cm. For the height, we decide to keep the sensor around the center height of the test model; therefore, in this case, the overall height should be approximately 25 cm + 25 cm = 50 cm above the base. From these simple calculations, the coordinate for the point that the robotic arm should move to is set as (X, Y, Z) = (−40 cm, 0 cm, 50 cm). Using this point coordinate, the software will calculate how to rotate the 6 joints of the robotic arm to move its tip to that point. Next, we need to calculate the coordinate where the robotic arm can scan the top of the test model. We are using a golden rule that lets the device lift to a 30° angle facing above the top of the test model. At this angle, the depth sensor will obtain a good coverage of the top. From Fig. 5b diagram, we are still trying to keep the scanning the same; therefore, we need to calculate the height at which the robotic arm must raise above the test model and the distance it must extend out from the base. Using trigonometry, we can find that $B = C \times \cos 30 \approx 52$ cm, $A = C \times \sin 30 = 30$ cm. Therefore, we can calculate the X-axis of the tip of robot arm at about (100 cm – 52 cm) = −48 cm, and the Z-axis should be 75 cm + 30 cm = 105 cm. The overall coordinate for this point should be (X, Y, Z) = (−48 cm, 0 cm, 105 cm).

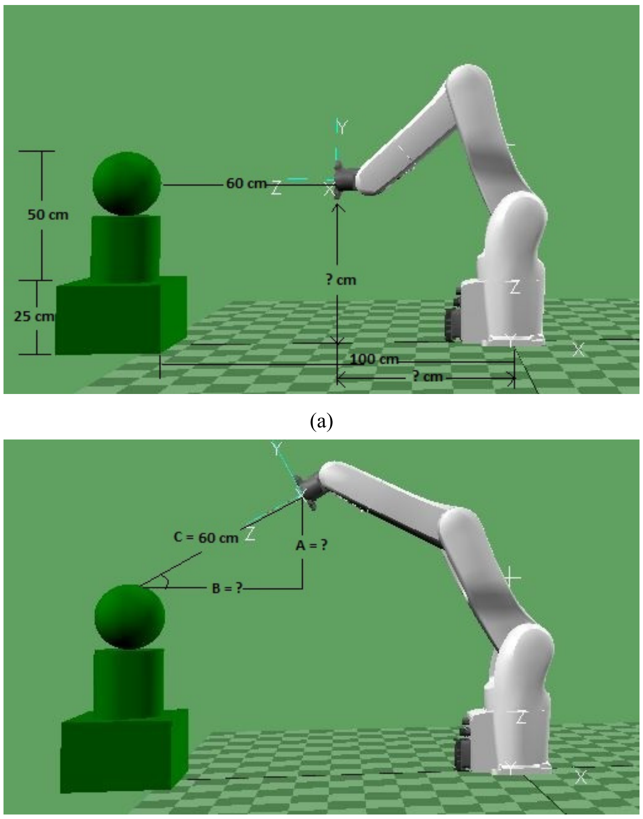

FIGURE 5. (a) Robotic arm at a scanning position scanning to the test model. (b) Robotic arm at a scanning position at the top of the test model.

With the two coordinates figured out, these corresponding robotic arm positions are saved into the robot's program. It is important to set an arc movement path for the robotic arm to move between these two positions so that the depth sensor on the test device always keeps the same scanning distance defined by the test case.

F. RUNNING THE AUTOMATION SYSTEM

In the actual testing phase, this automation setup design can be applied to scanning different kinds of test models with various testing conditions. Here, we have four basic test cases from our 3D reconstruction test plan (Please refer to Table 1). We need to scan these test models with different sizes, textures, scanning distances and light intensities. We check if the app will keep track of the testing models and complete the scanning with a good quality reconstructed 3D models. Likewise, we must catch app crashes or hang-ups during the stability test cases.

One of the biggest advantages of using the robotic arm for 3D scanning automation is its motion accuracy. To measure the robotic arm's accuracy, we attached a laser meter to the tip of the robotic arm where it holds the device. Since the test requires a constant specific scanning distance away from the test subject, we measured the distance on the side and the top scanning positions of the robotic arm to the test subjects, as shown in Figure 6.

TABLE 1. The four 3D reconstruction test cases.

Test Frequency	Test Subject/Setup	Test Description	Pass Criteria
On demand	Subject: Head model mannequin Lux: 500/300/100	Launch 3DR app at 60cm away from the mannequin. Reconstruct subject 3D model at different lux values.	No App crash Subject should be reconstructed successfully
On demand	Subject: Furry toy Lux: 500/300/100	Launch 3DR app at 60cm away from the furry toy. Reconstruct subject 3D model at different lux values.	No App crash Subject should be reconstructed successfully
On demand	Subject: Building model Lux: 500/300/100	Launch 3DR app at 60cm away from the building model. Reconstruct subject 3D model at different lux values.	No App crash Subject should be reconstructed successfully
On demand	Subject: Small china piece Lux: 500/300/100	Launch 3DR app at 45cm away from the china piece. Reconstruct subject 3D model at different lux values.	No App crash Subject should be reconstructed successfully

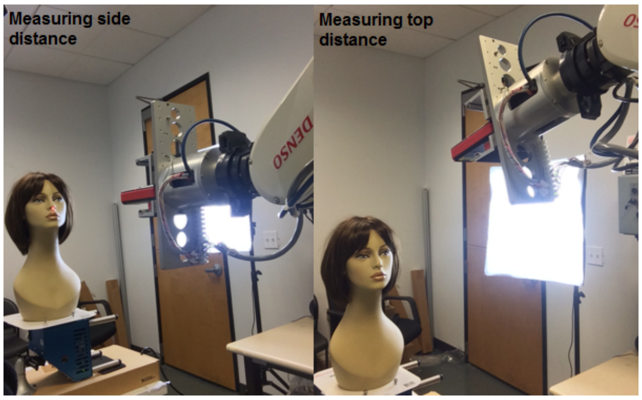

FIGURE 6. Using laser meter on robotic arm to measure the side and top distance.

TABLE 2. Distance measurement data on small, medium and large test subjects.

Distance Requirements	45cm	60cm	80cm
Small Test Subject			
Measured side distance	46.4cm	62.1cm	82.8cm
Measured top distance	44.3cm	58.4cm	77.7cm
Average Accuracy	98%	97%	97%
Medium Test Subject			
Measured side distance	43.6cm	58.4cm	78.0cm
Measured top distance	43.3cm	56.8cm	77.1cm
Average Accuracy	97%	96%	97%
Large Test Subject			
Measured side distance	42.5cm	58.9cm	77.7cm
Measured top distance	42.6cm	56.8cm	75.3cm
Average Accuracy	95%	97%	96%

We repeated this measurement on our standard small, medium and large size test subjects with scanning distances of 45 cm, 60 cm and 80 cm. The measured data and the calculated results are shown in Table 2.

The data in this table showed high consistency and accuracy across different sizes of the test subjects and all distances measured. The overall average accuracy calculated is approximately 96%, which proved that the robotic arm is a great choice when automating distance sensitive tests.

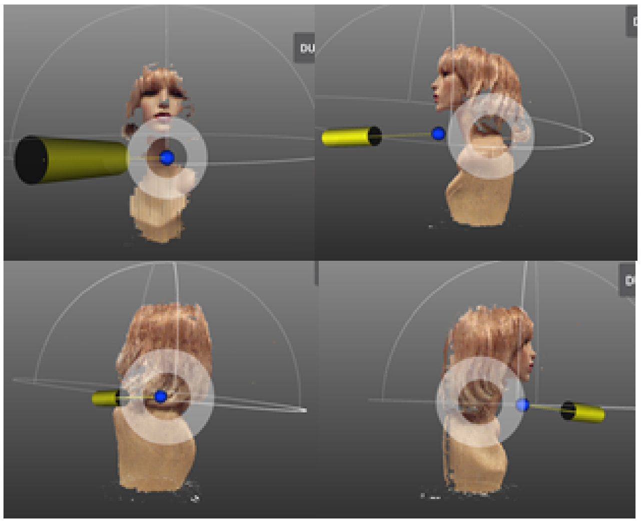

FIGURE 7. Screenshots of the 3D scan process at the 4 rotor stops.

With the robotic arm motion accuracy confirmed, we can start the 3D reconstruction automation testing listed in the test plan. The first test subject is the head model mannequin; it is set to be scanned at 60 cm distance with different light intensities. We took screenshots at every 90° rotor stop to check the tracking health status. This is shown in Fig. 7.

Using this automation system, we can successfully scan the rest of the test models and generate their 3D reconstruction models.

G. RESULTS

Since the robotic arm takes the scanning device scanning, the foundations of the 3D model are generated. Through the depth sensor, first a solid surface model is generated. Then, on top of that, the mesh of the model is created. The colors of the actual model captured by the camera sensor are filled-in on the mesh. From these processes, a full 3D model is generated.

During the 3D reconstruction test using robotic arm scans of these models, we made some observations.

- The 3D depth sensor required a minimum scanning distance between the sensor and the test subject. Some depth sensors are unable to detect objects in front of it if it is closer than a certain detection range.
- There will be limited textural details on the 3D model if it is scanning objects with too many details, such as something furry.
- Small test subjects are difficult to gather significant depth information. If the scanning is not close enough, it will lose the detailed textures. If the depth sensor gets close to capture the details, it will not be able to detect and scan the object due to range limitations.
- Avoid direct lighting on a reflective object surface. It will harm the scan quality due to excessive light reflected to the depth sensor.

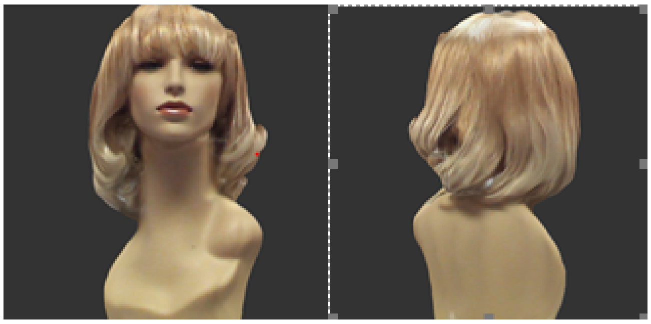

FIGURE 8. Captures of the reconstructed 3D model of the mannequin.

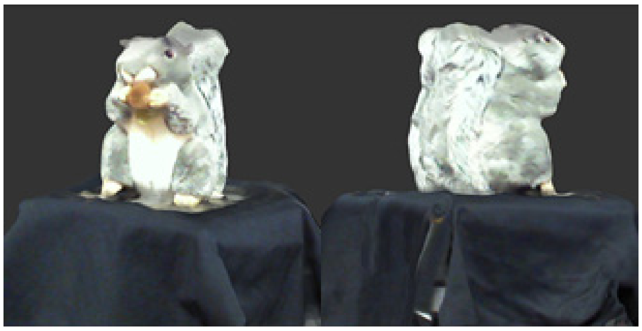

FIGURE 9. Captures of the reconstructed 3D model of the furry toy.

These are the 3D reconstruction scan results collected from running the automation and our evaluations.

1) HEAD MODEL MANNEQUIN

The automation captured almost every angle of the mannequin by following the guided path. The textures, details and colors are excellent in this 3D model that was generated from the scan.

2) FURRY TOY

On the furry toy model's scanning result, the overall 3D model shape is good as indicated in the snapshot of the complete model in Figure 9. The furry part is missing some texture on the top of the toy, including the two small ears. This is due to the known limitation regarding the high intensity of the details in the furry area without continuous scanning such as the body of the furry toy. Therefore, the ears are affected by the same colored risen tail as the background.

3) SMALL CHINA PIECE

On the small China piece, the scan captured all the color information well. However, there are some low definition and missing textures on some parts of the small China piece. The main reason for that is caused by the small size of the model, which makes it's hard for this depth sensor to detect and continuously track it. A more advanced depth sensor should perform better on this model.

4) BUILDING MODEL

The 3D reconstruction performed on the building model provided a relative good result. We can clearly see the windows and the textures. The automation did capture a small defect

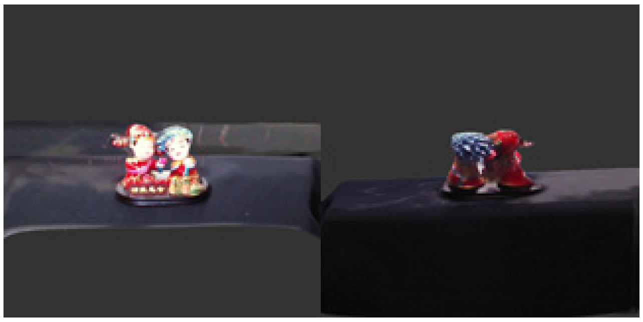

FIGURE 10. Captures of the reconstructed 3D model of the small China.

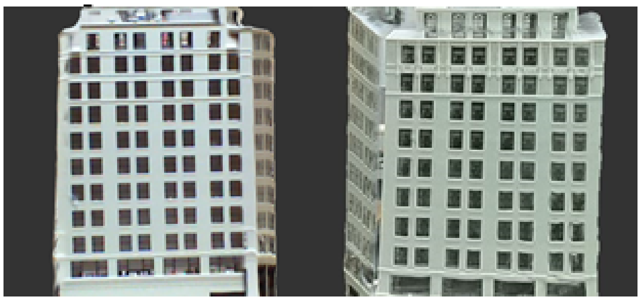

FIGURE 11. Captures of the reconstructed 3D model of the building model.

on the edge of this scan. This indicates that more tests need to be tested on this type of model to discover if it's a software performance issue on sharp edges or if it's due to the testing conditions (light source, scanning speed, magnitude of the shaking of this model, etc.).

Using the robotic arm, we can also perform a stability test on the 3D reconstruction software. We used the program to run the 3D scanning of the mannequin head model for 30 iterations, which took about an hour to complete. At the end of the stability test, we were able to see the successfully generated 3D model for each iteration and no app crash was detected. This is the first time that we can execute an end-to-end 3D scan stability test. It has proven the effectiveness of using the automation test system to test the stability of the 3D scanning use case, which could otherwise be extremely challenging to manually test.

V. CONCLUSION

The 3D reconstruction test automation is instrumental in creating a comprehensive test strategy to test functionality, performance and stability tests. 3D reconstruction tests have helped the engineering team support multiple software products and objectively benchmark the performance tests for multiple chipsets. The performance tests include key performance indicators, including latency measurement, memory and CPU profiling tests that help us evaluate the algorithm's performance.

3D reconstruction tests using the robotic arm lab have provided the advantage of performing panning of the device under different panning speeds and angles to the object. 3D reconstruction tests also involve testing the application program interface layer for algorithmic validation and then performing end-to-end validation using the android applications for 3D reconstruction. This approach has enabled us to thoroughly test and find any function level failures related to the 3D reconstruction algorithm before we perform the end-to-end validation using the inputs from the depth for stereo cameras.

VI. ADVANTAGES OF USING THE 3D RECONSTRUCTION TEST AUTOMATION

3D reconstruction testing involves the synchronization of a programmatically controlled robotic arm and a rotor. The distinctive feature of the 3D reconstruction test automation is that it provides an objective way to run automated tests and obtain precise results that can be benchmarked against several software builds.

The robotic arm has been programmed to execute test cases that pan the device for specific panning angles, panning speeds, and angular motions; this provides us with the ability to validate the algorithm by having the same orientation as the test subject.

It also uses an external depth sensor mounted onto the phone under test. This provides us the flexibility to change and add other external depth sensors and then validate the algorithm. The on-device test automation involves using Google Android's instrumentation method in android phones to launch intents, apply settings to an application, close the application, and other functions. We post-process the logs and the 3D model generated by the device and determine the results of the test case. This helps the on-device software validation of the 3D algorithms. We have added test cases in various software integration points. This has helped us catch issues related to the software or the 3D algorithms earlier in the software lifecycle.

The robotic arm has provided a framework for the addition of test scenarios. We can easily program the robotic arm to accommodate these new test scenarios. This includes new software changes or new 3D algorithm feature enhancements and performance optimizations.

REFERENCES

[1] J. L. de Bougrenet de la Tocnaye, B. Cochener, S. Ferragut, D. Iorgovan, Y. Fattakhova, and M. Lamard, "Supervised stereo visual acuity tests implemented on 3D TV monitors," *J. Display Technol.*, vol. 8, no. 8, pp. 472–478, Aug. 2012.

[2] C. A. Johnson and E. J. Casson, "Effects of luminance, contrast, and blur on visual acuity," *Optometry Vis. Sci.*, vol. 73, no. 12, pp. 864–889, 1995.

[3] J. S. Pointer, "Influence of selected variables on monocular, interocular, and binocular visual acuity," *Optometry Vis. Sci.*, vol. 85, no. 2, pp. 135–142, 2008.

[4] C. H. Y. Chiu and A. H. S. Chan, "Effect of screen contrast ratio and luminance level on visual lobe shape," in *Proc. Int. Multi-Conf. Eng. Comput. Sci. (IMECS)*, vol. 2. Hong Kong, Mar. 2008, pp. 19–21.

[5] M. T. M. Lambooija, W. A. Ijsselsteijna, and I. Heynderickx, "Visual discomfort in stereoscopic displays: A review," *Proc. SPIE*, vol. 6490, no. 1584, pp. 64900I–6490013, 2007. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6220259

[6] T. Shibata, J. Kim, D. M. Hoffman, and M. S. Banks, "The zone of comfort: Predicting visual discomfort with stereo displays," *J. Vis.*, vol. 11, no. 8, pp. 1–29, Jul. 2011.

[7] *Subjective Assessment of Stereoscopic Television Pictures*, document Rec. ITU-R BT.1438, ITU, 2000.

[8] T. Kanstrén, P. Aho, A. Lämsä, H. Martin, J. Liikka, and M. Seppänen, "Robot-assisted smartphone performance testing," in *Proc. IEEE Int. Conf. Technol. Pract. Robot Appl. (TePRA)*, May 2015, pp. 1–6.

[9] S. Elbaum and M. Diep, "Profiling deployed software: Assessing strategies and testing opportunities," *IEEE Trans. Softw. Eng.*, vol. 31, no. 4, pp. 312–327, Apr. 2005.

[10] A. Machiry, R. Tahiliani, and M. Naik, "Dynodroid: An input generation system for Android apps," in *Proc. 9th Joint Meeting Found. Softw. Eng. (ESEC/FSE)*, 2013, pp. 224–234.

[11] S. Anand, M. Naik, H. Yang, and M. Harrold, "Automated concolic testing of smartphone apps," in *Proc. ACM Conf. Found. Softw. Eng. (FSE)*, 2012, Art. no. 59.

[12] J. Jeon, K. Micinski, and J. Foster, "Symdroid: Symbolic execution for dalvik bytecode," Tech. Rep. CS-TR-5022, Jul. 2012. [Online]. Available: http://www.cs.umd.edu/~jfoster/papers/symdroid.pdf

[13] N. Mirzaei, S. Malek, C. Păsăreanu, N. Esfahani, and R. Mahmood, "Testing android apps through symbolic execution," in *Proc. Java Pathfinder Workshop (JPF)*, 2012, pp. 1–5.

[14] C. Cadar, D. Dunbar, and D. Engler, "KLEE: Unassisted and automatic generation of high-coverage tests for complex systems programs," in *Proc. 8th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*, 2008, pp. 209–224.

[15] P. Godefroid, N. Klarlund, and K. Sen, "DART: Directed automated random testing," in *Proc. ACM Conf. Program. Lang. Design Implement. (PLDI)*, 2005, pp. 213–223.

[16] J. C. King, "Symbolic execution and program testing," *Commun. ACM*, vol. 19, no. 7, pp. 385–394, Jul. 1976.

[17] R. C. Bryce, S. Sampath, and A. M. Memon, "Developing a single model and test prioritization strategies for event-driven software," *IEEE Trans. Softw. Eng.*, vol. 37, no. 1, pp. 48–64, Jan./Feb. 2011.

[18] A. Memon, M. E. Pollack, and M. L. Soffa, "Automated test oracles for GUIs," in *Proc. ACM Conf. Found. Softw. Eng. (FSE)*, 2000, pp. 30–39.

[19] A. M. Memon and M. L. Soffa, "Regression testing of GUIs," in *Proc. ACM Conf. Found. Softw. Eng. (FSE)*, 2003, pp. 118–127.

[20] X. Yuan, M. B. Cohen, and A. M. Memon, "GUI interaction testing: Incorporating event context," *IEEE Trans. Softw. Eng.*, vol. 37, no. 4, pp. 559–574, Jul./Aug. 2011.

[21] L. White and H. Almezen, "Generating test cases for GUI responsibilities using complete interaction sequences," in *Proc. 11th IEEE Int. Symp. Softw. Rel. Eng. (ISSRE)*, 2000, p. 110.

[22] X. Yuan and A. M. Memon, "Generating event sequence-based test cases using GUI runtime state feedback," *IEEE Trans. Softw. Eng.*, vol. 36, no. 1, pp. 81–95, Jan./Feb. 2010.

[23] D. Amalfitano, A. R. Fasolino, P. Tramontana, S. De Carmine, and A. M. Memon, "Using GUI ripping for automated testing of Android applications," in *Proc. 27th Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2012, pp. 258–261.

[24] T. Takala, M. Katara, and J. Harty, "Experiences of system-level model-based GUI testing of an Android application," in *Proc. 4th Int. Conf. Softw. Testing, Verification Validation (ICST)*, Mar. 2011, pp. 377–386.

DEBDEEP BANERJEE received the master's degree in electrical engineering from the Illinois Institute of Technology. He has approximately ten years of industry experience in the field of software/systems engineering. He has been with the Software Test Automation Team since the inception of the Computer Vision Project in Qualcomm Technologies Inc., USA, where he is currently the Software/Systems Development Engineer in test lead for the Computer Vision Project. He is a Senior Staff Engineer and an Engineering Manager with Qualcomm Technologies Inc. He is responsible for the test automation design, planning, development, deployment, code reviews, and managing the project. He is closely with the software/system teams and gathers test requirements for the project. He is involved in managing and developing software for the Computer Vision Lab using the robotic arm.

KEVIN YU is currently a Test Engineer with Qualcomm Technologies, Inc., USA, where he has contributed for test automation validation for continuous integration for computer vision algorithms. He has also validated computer vision engine features, such as image rectification, for the android software products.

GARIMA AGGARWAL is currently a Test Engineer with Qualcomm Technologies, Inc., USA, where he actively focused on MATLAB postprocessing modules for CV features and various other automation projects.