Received March 21, 2018, accepted April 18, 2018, date of publication April 23, 2018, date of current version May 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2829466

Hand Jitter Reduction Algorithm Software Test Automation Using Robotic Arm

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

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

ABSTRACT Hand jitter reduction (HJR) is an algorithm developed to offset the jitter effect caused by camera users. The hand jitter algorithm can be added and implemented in digital signal processing using hardware acceleration to execute these noise reduction algorithms. HJR involves the detection of noise in the camera images and the application of noise reduction algorithms to smoothen these noises. HJR tests are critical for validating the authenticity of the HJR algorithm. The challenge here is to be able to simulate similar motions to what a real-world user would experience while using the camera. We have inducted and programmed a robotic arm to simulate real-world user motions. We have developed software to simulate different motions, such as the generation of sine waves, and we also generate a randomized motion that is a combination of different motions, such as a sine wave, a cosine wave, and vertical, horizontal, and angular motions. In the robotic arm setup, the robot comprises six joints, and we can rotate the joints to generate a specific motion. We have developed an algorithm to find the locations of joints depending on the motion we need to simulate. For example, if we need a 30° rotation while the camera is at a specific location, we can calculate the joint value for specific joints in the robotic arm. The goal of HJR tests is also to categorize the results of the camera as having acceptable or non-acceptable results based on the induced motions. The test automation has immensely helped us to objectively benchmark the performance of the algorithm over several software builds. We have provided test results of computer vision use cases of camera panorama to show the effect of hand jitter on the quality of the software.

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

I. INTRODUCTION
Hand jitter reduction is an algorithm developed to offset the jitter effect caused by camera users. The hand jitter algorithm can be added and implemented in digital signal processing using hardware acceleration to execute these noise reduction algorithms. Hand jitter reduction involves the detection of noise in the camera images and the application of noise reduction algorithms to smoothen these noises.

Hand jitter reduction (HJR) tests are critical for validating the authenticity of the hand jitter reduction algorithm. The challenge here is to be able to simulate similar motions to what a real-world user would experience while using the camera. We have inducted and programmed a robotic arm to simulate real-world user motions. We have developed software to simulate different motions, such as the generation of sine waves, and we also generate a randomized motion that is a combination of different motions, such as a sine wave, a cosine wave, and vertical, horizontal, and angular motions. In the robotic arm setup, the robot comprises six joints, and we can rotate the joints to generate a specific motion.

We have developed an algorithm to find the locations of joints depending on the motion we need to simulate. For example, if we need a 30-degree rotation while the camera is at a specific location, we can calculate the joint value for specific joints in the robotic arm.

The goal of HJR tests is also to categorize the results of the camera as having acceptable or non-acceptable results based on the induced motions. The test automation has immensely helped us to objectively benchmark the performance of the algorithm over several software builds. We have provided test results of computer vision use cases of camera panorama to show the effect of hand jitter on the quality of the software.

II. MOTIVATION
The motivation for the design and development of the HJR simulation and automation using a robotic arm is to test

the HJR algorithm with scenarios that will not be covered by commonly used shake tables. We want to explore the possibilities of different kinds of hand jittering motions that a robotic arm can simulate, which will help us thoroughly test the HJR algorithm.

A. TEST REAL-WORLD USE CASES SINCE USERS CAN CAUSE JITTER WHILE USING CAMERA APPLICATIONS

The primary motivation for developing the hand jitter reduction test is to simulate the hand jitter generated by real-world users using the camera. We have designed the motions to be induced by the robotic arm that will simulate the shake or jitter a camera user may generate. The common motions include inducing jitter in the phone while shooting panoramic images.

B. ABILITY TO INTRODUCE REPEATABLE MOTIONS THAT CAN CREATE A COMBINATION OF SINE WAVES, COSINE WAVES, ETC

We have developed an algorithm to randomly generate predefined motions at specific time intervals while the tests are being executed. We combine these motions to simulate the randomness of real-world human hand jitter. These motions and the timing of their injection are recorded in a log file so that we can regenerate the same sequence of events over time to accurately reproduce the issues.

C. TEST THE RESILIENCE OF THE SOFTWARE ALGORITHM FOR NOISE CANCELLATION. PERFORM COMPETITIVE ANALYSIS OF THE NOISE CANCELLATION ALGORITHMS

We need to perform competitive analysis on the performance of the hand jitter reduction algorithm with other competitive products. Therefore, it is very important to understand the functionality of our in-house developed algorithm and what the tolerance limit is for motions and jitters. Then, we designed a test automation that will validate the motion types and force values that are both within and without the tolerable limits. First, the functional test cases are validated. Then, we check for the ability of the hand jitter reduction algorithm to correct for the jitter. If there are issues that cause the noise reduction algorithm to fail for the known intensity and type of motion, then we work with the systems/software teams to fix and resolve the issue.

D. TEST THE DSP OFFLOAD VERSUS NON-HARDWARE ACCELERATED SOLUTIONS AND QUANTIFY THE GAINS FROM USING HARDWARE ACCELERATED SOLUTIONS FOR USING THESE NOISE CANCELLATION ALGORITHMS

Since the jitter reduction can be computationally intensive, it is more power efficient to offload the computing on digital signal processors to achieve better performance. The challenge here is to automate the performance tests and prove how our hardware-accelerated solution is better. We can use the robotic arm lab automation to test the latency, CPU, and memory values of the software while the tests are executed with the hand jitter reduction algorithm. Therefore, we can repeatedly and accurately generate the performance test results.

E. ABILITY TO BENCHMARK THE CHANGES OF THE HAND JITTER REDUCTION ALGORITHM IN A CONTROLLED LIGHT TEST SETUP SO THAT WE CAN OBJECTIVELY TEST THE ALGORITHM CHANGES.

The test automation provides a reliable test setup in a lab environment with controlled light setup, precise motion generation capabilities and fixed force values to be generated for each motion. Therefore, due to this stable test setup, we can easily reproduce the software issues with the same sequence of motions, panning speeds, and panning distance from the test subject. In this way, we can execute the same tests for several software builds and benchmark the results to evaluate the algorithm's performance.

III. BACKGROUND AND RELATED WORK

It is important to design and develop automation for testing the hand jitter reduction algorithm. Robots have been used for testing precise movements and software [1]. Since we need precise angular movements to test hand jitter, the robotic arm provides us a mechanism to perform the camera shake.

Conventional approaches to quantitatively evaluate camera shaking include the analysis of the locus of a dot recorded on a captured image [2], the processing of the captured image using different test patterns [3], and the fixed test pattern method using a high-speed camera and feature points to evaluate camera jitter [4].

In the camera test automation, there needs to be end-to-end automation to launch the phone camera app, set the necessary settings and then perform the physical camera shake. The test automation should be able to launch the application for the testing example, including the face recognition application and the execution of test cases in which shake, or jitter is physically added.

To automate the phone software (using operating systems such as Android), we need tools and software developed to test these phone applications [6]. These include running automated regressions and stability tests [7].

Android applications inject events when users press GUI (graphical user interface) buttons on the phone. The suitable models of the system or sub-system to be tested include event-flow graphs, event-interaction graphs, or finite state machines [9], [10], [12]. When the user uses applications on the phone, traces can be collected on the event sequence exercised by the user. These session traces can be used for deriving test cases [8] or can be based on GUI rippers [11]. Application crawlers [13] can be used that automatically deduce possible sequences of events that can be translated into test cases.

Google Android provides a tool called "Monkey" that is a random key presser [14]. Since this tool is widely used, it is regarded as the current state of practice for automated software testing in Android [15]. It uses a standard, simple-but-effective, default test oracle [16] that regards any input that leads to a crash to be a fault-revealing test sequence.

Exploratory testing is ''simultaneous learning, test design, and test execution'' [17] that can be cost-effective and is widely used by industrial practitioners [18]–[20] for general testing. However, it is particularly underdeveloped for mobile app testing [21], [22].

For the design and development of testing Android-based applications GUI, application programming interface(API) level or model-based approaches have been taken [23]–[28].

App event sequences can be generated from models that are manually constructed or obtained from project artifacts, such as code, XML configuration files or UI execution states. For example, AndroidRipper [24] builds a model using a depth-first search over the user interface in the phone software.

Some of the techniques require detailed app information, such as source code [29]. Additionally, there is a need for general UI models [30] and interface and activity transition models [31], [32].

The usage of a robotic arm for software testing is prevalent as we can precise use a robot to repeat a specific test sequence accurately [33]. The robots have been used for performing graphical user interface testing [34] etc. Black box testing involves testing of a software program without knowing the internal implementation of the software functions and its building blocks. The software program can be tested in a black box technique if the inputs and the expected output is known. A robot setup can be efficiently used for software testing as it can precisely repeat the test sequences and the test results can then be analyzed [35]–[40].

Therefore, we have a reference to the tools and techniques of software automation of phone software applications that can be used to execute tests on the phone. These tests can be executed when the phone is shaken during the hand jitter algorithm testing.

IV. SOLUTION

In the video encoding for hand jitter reduction testing, a shake table is commonly used for generating motions and testing digital image stabilization algorithms. In image stabilization algorithms, the algorithm detects and eliminates the jitter by comparing the frames and identifying the frames that have jitter. It can apply algorithms to smooth the jitter in the recorded video. Similarly, in the hand jitter reduction (HJR) algorithm tests, we can use a robotic arm to generate very precise and repeatable motions to the device so that we can evaluate the hand jitter reduction algorithm's performance under different motions.

A. USE CASES TO BE TESTED FOR HAND JITTER REDUCTION ALGORITHM TESTS

The use cases that can be tested with HJR include camera preview, video recording, and computer vision use cases such as camera panorama, face recognition, and 3D reconstruction. Any use cases that use live frames from the camera sensor can be used with hand jitter reduction algorithms since it can help smooth the issues and errors due to the motion.

B. QUANTIFICATION OF THE IMPACT OF HAND JITTER (SHAKE) ON THE COMPUTER VISION ALGORITHMS TESTS

We work closely with the computer vision engineering team to understand the capabilities of the hand jitter reduction algorithm and perform tests with functional and adversarial conditions. The algorithm can handle and eliminate jitter for the specific threshold of motion, and the algorithm will fail to detect it beyond the threshold. Therefore, we understand this threshold concerning angular motion and then design test cases to test specific panning speed, angular motions, and other aspects. Tests that are beyond the acceptable range of motion are considered as adversarial tests, and we do not fail these tests since the expected result is a failure here. We do check if the algorithm can gracefully handle the adversarial tests and not crash the software.

C. TEST SETUP

We have a dedicated lab for executing HJR use case automation. This lab system contains three essential components, including programmable light sources, the test subjects setup and a DENSO robotic arm setup.

We have installed adjustable light sources that can control the brightness and color of the test environment. These light sources can be programmed and controlled by a central PC remotely. Therefore, we can add light conditions to the HJR test with different lux values. In video recording, lighting condition is a key factor of the FPS of recorded video. It will affect the HJR performance as well. In a darker room, some of the captured video may be too dark to be correctly analyzed by the HJR algorithm and resulted in HJR performance drop at certain lux values. Therefore, it is necessary to have a controllable light source to effectively find issues in HJR testing.

In this lab, we have multiple large posters and 3D models installed as our test subject setup. They serve as our test contents and simulating what the software will be applied in real-world environment. They are placed in fixed positions so that we can program the robotic arm to take the test device to different test subjects designed for each test case. Every time, the device will be moved precisely to the same point and perform the same hand jitter movement in the same test case. Therefore, we can easily reproduce any previous issue, which helps developers debug these issues. The robotic arm is the most critical piece of our lab setup. We are using the DENSO VS-Series six-axis articulated robot. There are some clear advantages of using this robotic arm in our test automation. It offers high precision that is important when we require it to rotate the device to a certain angle. It keeps the test device at a precise distance from the test subjects. The compact design of the robotic arm helps us save lab space. The six-axis motion design makes it highly flexible. It can perform continuous motions and difficult angular motions that the human arm cannot do. Figure 1 shows the test setup.

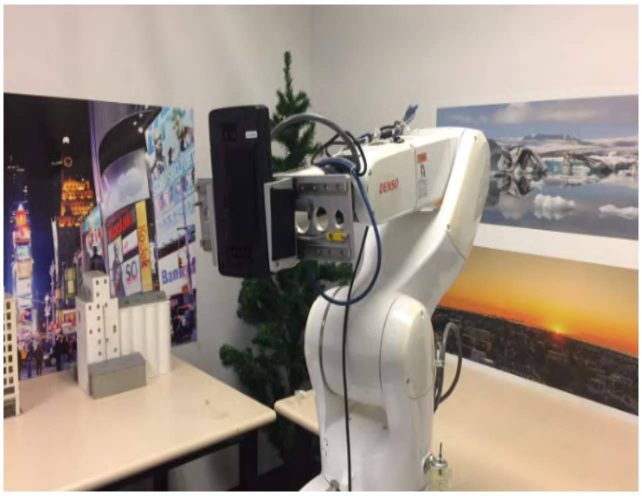


FIGURE 1. **Robotic arm lab Setup for executing the hand jitter reduction verification tests.**

D. HAND JITTER TEST AUTOMATION ALGORITHM

The test automation algorithm for hand jitter reduction using the robotic arm is illustrated in Figure 2.

E. THE ADVANTAGE OF USING A ROBOTIC ARM SETUP FOR TESTING HAND JITTER REDUCTION TESTS

The advantage of using the robotic arm setup for testing hand jitter reduction is that we can simulate injections of jitter at various points of the camera's motion. For example, if a user is running a camera use case, such as camera preview, jitters can be introduced. The jitters can be injected in a repeatable fashion at specified time intervals, say every 10 milliseconds. The goal of running these tests is to check the quality of the camera use cases while these jitters are induced. We can program these jitters in the robotic arm setup.

Statistically, a software program can generate these injections of jitters in the robotic arm at either randomized or a repeatable fixed instance of time. These will help to simulate real-world scenarios where the user of a camera is traveling in a vehicle that may have a jerk while turning, braking, or in another action. We will evaluate the effect of these jitters in the camera application software.

We have worked closely with the computer vision systems and software team to calibrate the effect of these jitters and divide the test cases as functional and adversarial tests. Functional tests are the ones where the software algorithm can conceal the effect of these jitters, whereas the adversarial tests are higher than the permissible level where the software can conceal these jitters.

F. GENERATING A COSINE WAVE MOTION USING ROBOTIC ARM

To simulate hand jittering in our computer vision testing, we need to generate different kinds of motions that a human hand is likely to exhibit while holding a phone. For example, in the panorama use case, one of the motions that a human hand would likely perform while taking a long panoramic picture is a sine or cosine wave-like motion. People take panoramic pictures by holding the camera with their arms stretched out and rotate around the area on which they stand. During this panning motion, it is very likely that their arms go slightly up and down to adjust the camera's level. In this case, the panning motion is similar to a sine/cosine wave.

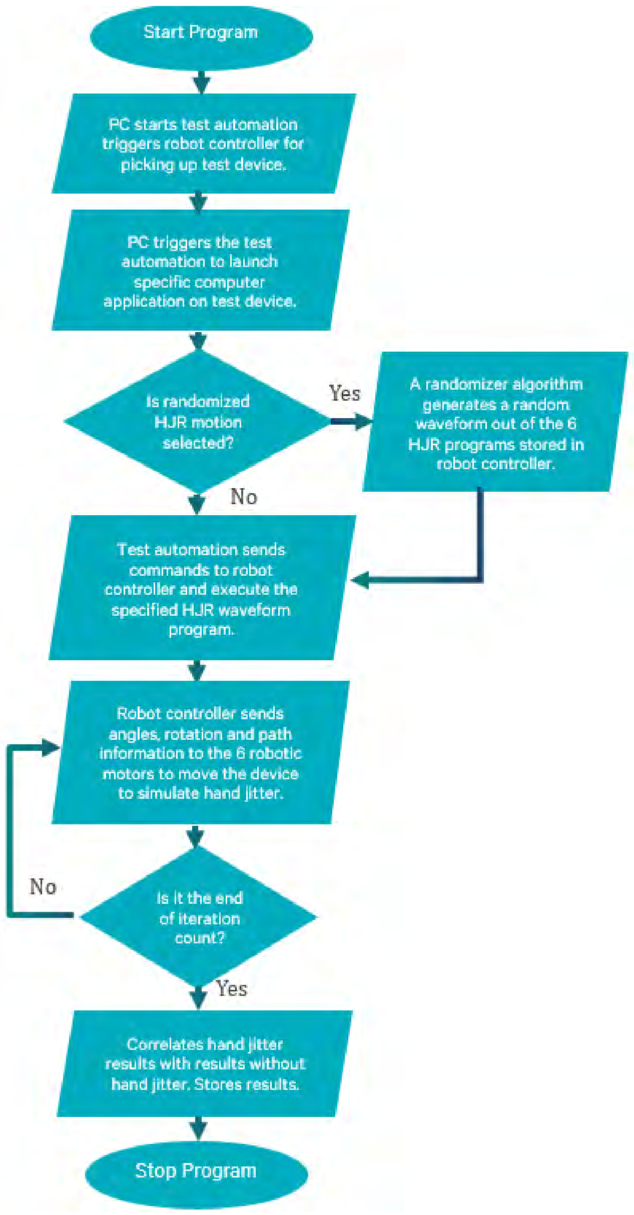

Start Program
PC starts test automation triggers robot controller for picking up test device.
PC triggers the test automation to launch specific computer application on test device.
Is randomized HJR motion selected?
Yes
A randomizer algorithm generates a random waveform out of the 6 HJR programs stored in robot controller.
No
Test automation sends commands to robot controller and execute the specified HJR waveform program.
Robot controller sends angles, rotation and path information to the 6 robotic motors to move the device to simulate hand jitter.
No
Is it the end of iteration count?
Yes
Correlates hand jitter results with results without hand jitter. Stores results.
Stop Program

FIGURE 2. **Hand Jitter Reduction Test Automation Algorithm.**

We have designed a program using WINCAPS 3 software to perform such a cosine wave using the robotic arm. The concepts related to creating cosine wave motions include the following:

1. Create multiple periods of cosine wave to exercise hand jittering removal algorithm.
2. Create irregular cosine waves that vary in amplitude and period length to simulate a real-life scenario.
3. Provide smooth transitions between waves.

In WINCAPS 3 software, sine or cosine wave paths can be designed using ARCHMOVE commands. First, we define the initial starting position P1 and destination position P2 in the ARCHMOVE parameters. In this case, we want to start right from P1 and end at P2 for a continuous arc motion. Then, it needs to define the arc shape and amplitude by assigning two PASS positions P3 and P4. The motion path will approach these points and pass by them in an arc shape. We combine the parameters P1, P2 and P3, P4 and feed them into the ARCHMOVE command, which creates a defined arc path. Figure 3 illustrates this.

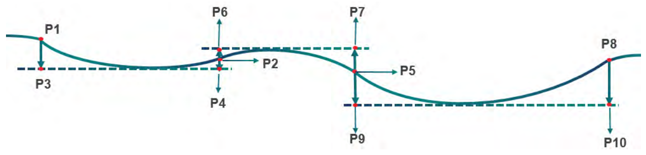


FIGURE 3. Cosine wave motion design diagram.

We repeat the same arc creation process multiple times with the defined position parameters P5, P6, P7, P8, P9, P10 to create an irregular, smooth cosine-like path. We simulate this arc motion in the WINCAPS3 software 3D simulator, where it rotates starting from facing one poster on the left and ends facing the second poster on the right. Please refer to Fig. 4.

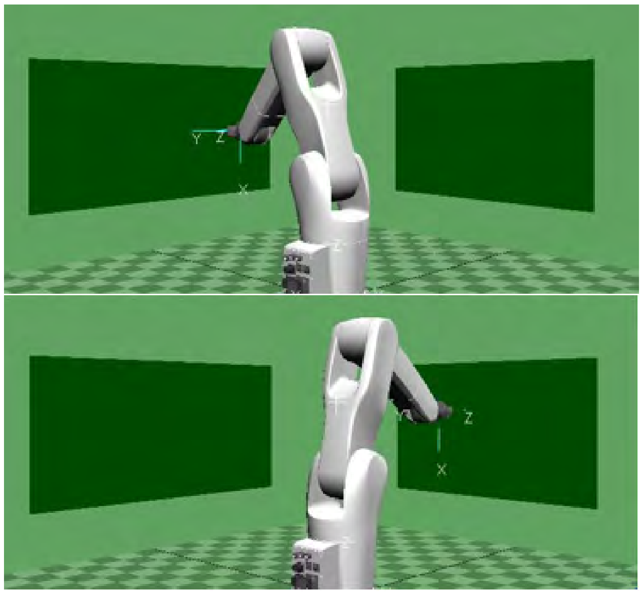


FIGURE 4. 3D simulation of robotic arm moving from P1 to P8.

During the rotation, the tip of the robotic arm draws the cosine path, scanning through both posters. This is a use case for hand jittering scenario in camera panorama testing. We can use this program to test how well the panorama software's HJR algorithm removes the jitter caused by this cosine motion.

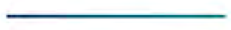
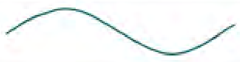
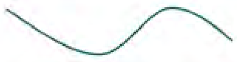
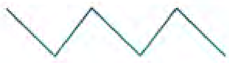
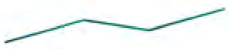
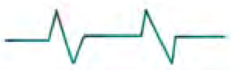
G. INTRODUCTION OF GENERATING RANDOMNESS IN MOTION

It is very important to introduce randomness in the motions generated for testing hand jitter. The goal of these tests is to simulate the hand jitter that real-world users generate using a smartphone camera. Typically, we can create the randomness in the software by creating a hash table that has keys linked with values as the type of motions.

A randomizer algorithm generates a random number between the ranges of the keys after definable time intervals. For example, at the start of the test, it generates a random number from 1 to 6. Now, this random number is associated with the value of the key. The corresponding value for the key is a sine wave. The robotic arm will start loading the sine wave program from its program list, and it will move the test device in a sine wave. Then, the script will again generate another random number after the previous hand jitter program completes.

We have programmed 6 total waveform programs that we are likely to encounter during human hand jittering, as shown in Table. 1. These waveforms cover most of the hand jittering scenarios, such as people taking a video or a panoramic picture while walking, riding in a moving vehicle or sitting on a boat. Each of them is assigned an index key. When we run the Randomizer program, it will generate different motions over time that exercise most of the hand jittering motions using the robotic arm.

TABLE 1. Waveform table with associating index numbers.

Index	Waveform	Wave Graph
1	Straight	
2	Sine	
3	Cosine	
4	Triangle	
5	Ramp up	
6	Pulse	

H. CONTROLLED TESTING SPEED

The robotic arm system offers adjustable movement speed options. For each test scenario, we can run the designed hand jitter movements at a certain speed. The higher the speed, the higher the change rate between frames. Therefore, it is more challenging for the HJR algorithm to render a smooth picture or video at higher speed. We have also designed a robotic arm program to change the movement speed from point to point during the hand jittering motion. This change of velocity creates acceleration, which is a key scenario to test real-life hand jitter since user's hand jitter speed does not stay constant. Additionally, by starting from a certain scale of acceleration, we do expect HJR performance to decrease. This test information will be critical for benchmarking purpose.

V. RESULTS

As mentioned in the previous section, we have created a robotic arm program that produces hand jitter in a cosine wave motion while taking a panoramic picture. Before we can get meaningful test results, we must know how the robotic arm actually moves, including the speed, the path and the magnitude of the cosine wave. Therefore, we ran the program on the robotic arm and placed laser measuring equipment on the tip of the robotic arm to measure the altitude of the tip to the ground, as shown in Fig. 5.

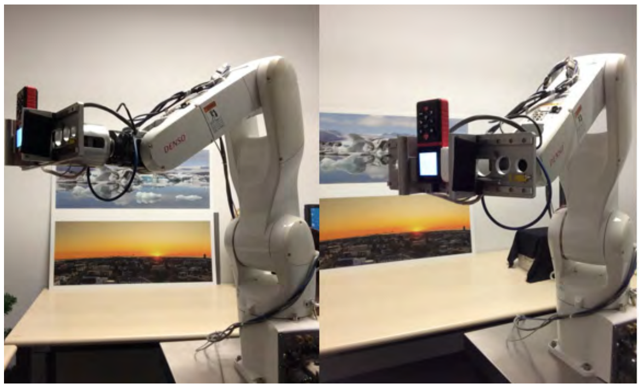


FIGURE 5. Laser equipment setup on the robotic arm for altitude measurement.

The laser was programmed to take a reading every 1 second. The robotic arm finished the cosine panning motion in 21 seconds moving from one end to the other. The data was collected as shown in Table 2.

TABLE 2. Device altitude data collected over 21 seconds.

Measured device altitude for every second											
Time (Sec)	0	1	2	3	4	5	6	7	8	9	10
Altitude (m)	1.543	1.543	1.542	1.535	1.522	1.515	1.509	1.513	1.519	1.527	1.525
Time (Sec)	11	12	13	14	15	16	17	18	19	20	21
Altitude (m)	1.52	1.507	1.494	1.485	1.487	1.491	1.497	1.507	1.519	1.525	1.525

From this table, we can see the device is moving up and down approximately 1.5 meters above the ground. This is

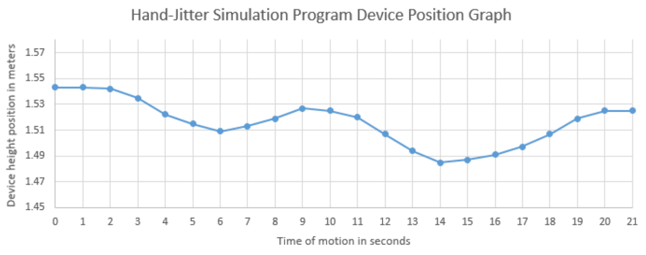
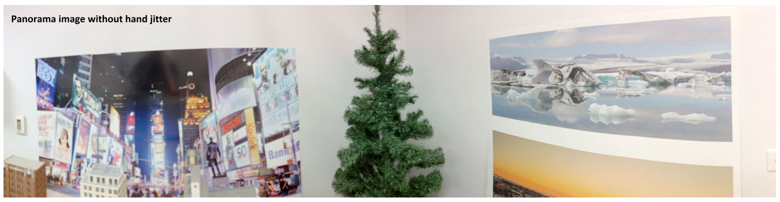


FIGURE 6. Hand jitter simulation device position graph.

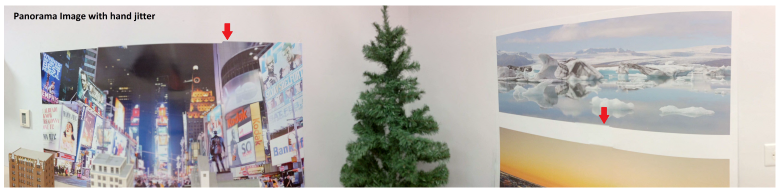
the height at which most people hold their cameras when taking a panoramic picture. The absolute difference between the highest and the lowest altitude is 5.8 centimeters, which is a small magnitude of hand jitter in a panoramic path. We plot the data into a graph and the graph does show the path forms a cosine wave, as shown in Figure 6. Note that there are two cosine periods in this designed path, which means there are two local minimum points. We expect to see two peaks in terms of image shift on the panoramic image taken.

After proving the robotic arm moves the test device in a cosine motion as we designed. We started testing the HJR algorithm in panorama software with the robotic arm. At the first run, we tested it without the cosine hand jitter motion by running the robot straight from the start point to the destination. The result panoramic image is shown in Figure 7a. We can see in the picture that all the key frames are stitched together smoothly. Then, on the second run, we introduced hand jitter by testing with the cosine wave robotic arm program. The resulting picture is shown in Figure 7b. We did observe two abnormal curvatures introduced by the hand jitter motion with upward shifts located approximately at the two local minimum points of the cosine wave (indicated with red arrows), while the rest of the image are stitched smoothly. This is due to the device was at the lowest altitude at these two points. The poster images captured at these points appear to be higher positioned compare to the adjacent key frames. When stitching these key frames at different levels the abnormal curvatures are likely to appear.

In order to quantitatively analyze the panorama HJR result images, an objective image processing method is needed. We have came up with a post-processing solution that compares four sets of images. First, we run the robotic arm program without hand jitter. The automation controls the test device to take camera snapshot every one second during the panning process. Then we run the program with hand jitter and take camera snapshots for every second as well. The two sets of snapshot images are being paired by the time-in-motion. And the only image difference in each pair should came from the hand jitter effect. The two sets of snapshot result images are processed using Matlab to find a similarity score between each pair without considering color differences. This should capture the hand jitter effect magnitude in term of image differences. And it should be proportional to the cosine hand jitter waveform. After that, multiple sections of images are extracted from the two panoramic result images



(a)



(b)

FIGURE 7. (a) The panoramic image captured without hand jitter. (b) The panoramic picture captured with hand jitter.

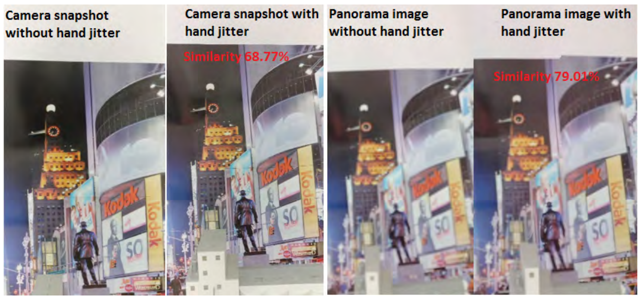

Camera snapshot without hand jitter
Camera snapshot with hand jitter
Similarity 68.77%
Panorama image without hand jitter
Panorama image with hand jitter
Similarity 79.01%

FIGURE 8. Similarity comparison between camera snapshot and panorama images with & without hand jitter at 6th second.

that we captured. These pairs of extracted images should have the same dimensions and match each set of the camera snapshot images accordingly. Finally, we use the same Matlab program to find the similarity score of each pair of the extracted panoramic images. The comparison between two pairs at the 6th second are illustrated in Fig. 8. Using this method, we can have a reference to compare the hand jitter effect to the result images, and therefore quantitively find out the effect of the hand jitter reduction algorthim. With the benefit of the precision and consistency using the robotic arm, we can get relatively accurate results.

After finding the similarity scores of the four sets of images, the similarity scores are converted into similarity differences in percentage for better showing the hand jitter effect to the images. The similarity difference data of the twenty seconds of hand jitter motion in this case are presented in table 3 below.

As we can see from this table, there are some image noises from this image comparison solution which resulted in between 5% to 10% image similarity difference. For example, at 1st and 20th second, there are not much hand jitter happening. This is due to the minor difference in image capture timing and slight displacement between compared images. However, we can clearly see the panorama hand jitter reduction algothrim taking effect by aligning the key

TABLE 3. HJR result comparison table for camera snapshot and panorama.

HJR Comparison Result Table										
Time (sec)	1	2	3	4	5	6	7	8	9	10
Panorama Image Similarity Difference	6.70%	8.60%	10.89%	17.19%	19.46%	20.99%	18.13%	12.73%	7.06%	5.85%
Snapshot Image Similarity Difference	8.47%	10.02%	13.66%	21.79%	27.70%	31.23%	24.75%	16.28%	9.96%	9.48%
HJR Delta	1.77%	1.42%	2.77%	4.60%	8.24%	10.24%	6.62%	3.55%	2.90%	3.63%
Time (sec)	11	12	13	14	15	16	17	18	19	20
Panorama Image Similarity Difference	7.49%	12.44%	18.68%	22.96%	21.87%	18.57%	14.49%	8.90%	7.49%	7.77%
Snapshot Image Similarity Difference	14.41%	17.16%	26.70%	34.83%	32.62%	28.29%	19.26%	10.30%	8.43%	7.60%
HJR Delta	6.92%	4.72%	8.02%	11.87%	10.75%	9.72%	4.77%	1.40%	0.94%	-0.17%

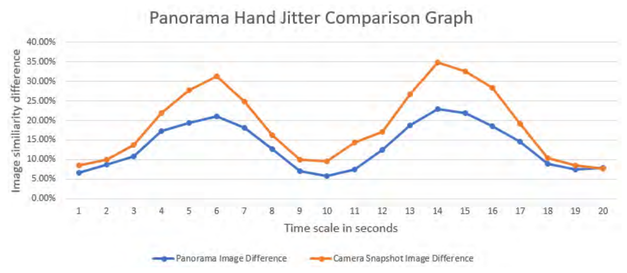


FIGURE 9. Panorama HJR comparison graph with camera snapshot.

frames properly. After converting the data on table 3 into a graph shown in Fig. 9, we can see the two peaks in similarity difference at the 6th second and 14th second, these are also the same time where the cosine form hand jitter reached its local minimum in altitude which is an expected result. At these two peaks the panorama images similarity difference is much lower than the camera snapshot. This means the displacement in panorama image is smaller than the actual displacement of the device when hand jitter happens. And at

TABLE 4. Advantages of using the robotic-arm-based hand jitter reduction algorithm test automation.

Topics	On Device Test Automation with Robotic Arm	Test Setups for Testing Hand Jitter
Angular movements to create hand jitter.	The Denso robotic arm has provided us with the ability to perform angular movements up to 360 degrees. It has 6 robotic motors that provide precise angular movements and can be programmed.	Programmatically, it may provide an interface to change the angular movements for rotating and panning the phone under tests at different speeds.
Ability to create different hand jitter permutations/ combinations.	We used 6 different programs with different combinations of waves varied with time. These include a straight line and cosine, sine, triangle, ramp-up, and pulse waves. These can be altered based on the timing.	The setup may not be flexible enough to add new test scenarios or programs for simulating new hand jitter sequences.
Test scalability and the ability to add new hand jitter scenarios programmatically.	The test automation supports a framework that makes it easy to add new automated test cases into it. In this case, it is easy to add new test scenarios with randomized sequences of jitter as a combination of ramp-up and pulse waves, etc.	New randomized programs may not be added as a function of time to create new permutations of hand jitter sequences.
On device test automation support for executing hand jitter reduction algorithm tests.	On device phone test automation is present, which works with the robotic arm automation. The computer vision-based application of the mobile phone software can be launched, settings can be applied, test can be executed, and the app can be closed.	The test setup may not provide addition on-device test automation.

the time hand jitter is minimal, the similarity differences are getting very close. This proves that our solution are able to effectively identify and quantify the HJR effect on images.

VI. COMPARISONS WITH OTHER TEST SYSTEMS

Table 4 shows the advantages of using the robotic-arm-based setup for hand jitter reduction.

The Table 3 compares the robotic arm-based test automation capabilities of the computer vision use case testing with hand jitter reduction algorithm enabled with other test systems. Computer vision use cases like camera panorama which involves complex algorithms for taking snapshots of reference frames based on motion vector changes and stitching all the reference frames to generate a panoramic image is subjected to tests for measuring the effect of hand jitter. The hand jitter reduction algorithm effect on the computer vision usecases can be quantified and with this test setup. The test results are consistently benchmarked across multiple software product lines and we can programmatically scale the test scenarios in the robotic arm setup.

VII. CONCLUSION

The test results illustrate that we can simulate the jittery camera movements of real-world users. We have performed tests to evaluate the quality of the computer vision use cases like camera panorama when hand jitter is induced. We have programmed the device under test to capture snapshots of images at specific interval of times and then induce hand jitter motions using the robotic arm. We use correlation to compare the images with and without the hand jitter induction. We infer from the results that the image dissimilarities between same usecases with and without jitter is directly proportional to the amount of jitter induced. The software algorithms for image stabilization tries to stabilize the effect of hand jitter till a specific motion level and after this we see distortions in the captured images due to hand jitter. We have developed motion sequences to benchmark these test results across multiple software product and also categories motion level as positive and negative test cases. Positive test cases match to the expectation of the software to eliminate the jitter by apply image stabilization and motion compensation algorithms. The negative test cases are developed to test the software capabilities to handle hand jitter to a higher level of motion changes. These jitter values are above the permissible level and it tests the software application resilience to crash etc.

We also observe that computer vision usecases like camera panorama is efficient in applying motion compensation and stitching the reference frames to produce better results even when the hand jitter is induced. However, camera snapshot usecases with the same jitter values and test subject may show distortions when hand jitter to a specific value is applied. The robotic arm test automation has provided us the ability to scale the test automation deployment over multiple software products. The hand jitter reduction algorithms are tested and benchmarked over multiple software products.

REFERENCES

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

[2] K. Hayashi, M. Tanaka, H. Kusaka, and H. Hashi, "New approach on multi-axial analysis of camera shake," in *Dig. Tech. Papers Int. Conf. Consumer Electron. (ICCE)*, 2010, pp. 39–40.

[3] F. Xiao, A. Silverstein, and J. Farrell, "Camera-motion and effective spatial resolution," in *Proc. Int. Congr. Imag. Sci.*, 2006, pp. 33–36.

[4] K. Nishi and R. Ogino, "3D camera-shake measurement and analysis," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2007, pp. 1271–1274.

[5] E. M. Or and D. Pundik, "Hand motion and image stabilization in hand-held devices," *IEEE Trans. Consum. Electron.*, vol. 53, no. 4, pp. 1508–1512, Nov. 2007.

[6] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real challenges in mobile app development," in *Proc. ACM /IEEE Int. Symp. Empirical Softw. Eng. Meas.*, Oct. 2013, pp. 15–24.

[7] D. Amalfitano, A. R. Fasolino, and P. Tramontana, "A GUI crawling-based technique for Android mobile application testing," in *Proc. IEEE 4th Int. Conf. Softw. Test., Verification Validation Workshops*, Mar. 2011, pp. 252–261.

[8] D. Amalfitano, A. R. Fasolino, and P. Tramontana, "Rich Internet application testing using execution trace data," in *Proc. 2nd Int. Workshop Test. Techn. Experim. Benchmarks Event-Driven Softw. (TESTBEDS)*, Apr. 2010, pp. 274–283.

[9] F. Belli, C. J. Budnik, and L. White, "Event-based modelling, analysis and testing of user interactions: Approach and case study," *Softw. Test., Verification Rel.*, vol. 16, no. 1, pp. 3–32, Mar. 2006.

[10] A. Marchetto, P. Tonella, and F. Ricca, "State-based testing of Ajax Web applications," in *Proc. Int. Conf. Softw. Test., Verification Validation*, 2008, pp. 121–130.

[11] A. M. Memon, L. Banerjee, and A. Nagarajan, "GUI ripping: Reverse engineering of graphical user interfaces for testing," in *Proc. 10th Working Conf. Reverse Eng. (WCRE)*, 2003, pp. 260–269.

[12] A. M. Memon and Q. Xie, "Studying the fault-detection effectiveness of GUI test cases for rapidly evolving software," *IEEE Trans. Softw. Eng.*, vol. 31, no. 10, pp. 884–896, Oct. 2005.

[13] A. Mesbah and A. van Deursen, "Invariant-based automatic testing of AJAX user interfaces," in *Proc. Int. Conf. Softw. Eng. (ICSE)*, 2009, pp. 210–220.

[14] K. Mao, M. Harman, and Y. Jia, "Sapienz: Multi-objective automated testing for Android applications," in *Proc. 25th Int. Symp. Softw. Test. Anal. (ISSTA)*, 2016, pp. 94–105.

[15] R. Mahmood, N. Mirzaei, and S. Malek, "EvoDroid: Segmented evolutionary testing of Android apps," in *Proc. ESEC/FSE*, 2014, pp. 599–609.

[16] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 507–525, May 2015.

[17] A. Abran *et al.*, *Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2004.

[18] J. Bach, "Exploratory testing explained," v.1.3 4/16/03. [Online]. Available: http://www.satisfice.com/articles/et-article.pdf

[19] J. Itkonen and K. Rautiainen, "Exploratory testing: A multiple case study," in *Proc. ESEM*, Nov. 2005, pp. 84–93.

[20] C. Kaner, J. Bach, and B. Pettichord, *Lessons Learned in Software Testing*. New York, NY, USA: Wiley, 2008.

[21] J. Itkonen, M. V. Mantyla, and C. Lassenius, "How do testers do it? An exploratory study on manual testing practices," in *Proc. ESEM*, vol. 9. Oct. 2009, pp. 494–497.

[22] J. Itkonen, M. V. Mantyla, and C. Lassenius, "The role of the tester's knowledge in exploratory software testing," *IEEE Trans. Softw. Eng.*, vol. 39, no. 5, pp. 707–724, May 2013.

[23] D. Amalfitano, A. R. Fasolino, P. Tramontana, B. D. Ta, and A. M. Memon, "MobiGUITAR: Automated model-based testing of mobile apps," *IEEE Softw.*, vol. 32, no. 5, pp. 53–59, Sep. 2015.

[24] 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. ASE*, 2012, pp. 258–261.

[25] T. Azim and I. Neamtiu, "Targeted and depth-first exploration for systematic testing of Android apps," in *Proc. OOPSLA*, 2013, pp. 641–660.

[26] W. Choi, G. Necula, and K. Sen, "Guided GUI testing of Android apps with minimal restart and approximate learning," in *Proc. OOPSLA*, 2013, pp. 623–640.

[27] S. Hao, B. Liu, S. Nath, W. G. J. Halfond, and R. Govindan, "PUMA: Programmable UI-automation for large-scale dynamic analysis of mobile apps," in *Proc. MobiSys*, 2014, pp. 204–217.

[28] W. Yang, M. R. Prasad, and T. Xie, "A grey-box approach for automated GUI-model generation of mobile applications," in *Proc. FASE*, 2013, pp. 250–265.

[29] S. Anand, M. Naik, M. J. Harrold, and H. Yang, "Automated concolic testing of smartphone apps," in *Proc. ESEC/FSE*, 2012, pp. 59:1–59:11.

[30] C. S. Jensen, M. R. Prasad, and A. Møller, "Automated testing with targeted event sequence generation," in *Proc. ISSTA*, 2013, pp. 67–77.

[31] N. Mirzaei, J. Garcia, H. Bagheri, A. Sadeghi, and S. Malek, "Reducing combinatorics in GUI testing of Android applications," in *Proc. ICSE*, 2016, pp. 559–570.

[32] N. Mirzaei, S. Malek, C. S. Păsăreanu, N. Esfahani, and R. Mahmood, "Testing Android apps through symbolic execution," *SIGSOFT Softw. Eng. Notes*, vol. 37, no. 6, pp. 1–5, 2012.

[33] D. Banerjee, K. Yu, and G. Aggarwal, "Robotic arm based 3D reconstruction test automation," *IEEE Access*, vol. 6, pp. 7206–7213, Jan. 2018, doi: 10.1109/ACCESS.2018.2794301.

[34] K. Mao, M. Harman, and Y. Jia, "Robotic testing of mobile apps for truly black-box automation," *IEEE Softw.*, vol. 34, no. 2, pp. 11–16, Mar./Apr. 2017.

[35] V. Garousi and M. Felderer, "Worlds apart: Industrial and academic focus areas in software testing," *IEEE Softw.*, vol. 34, no. 5, pp. 38–45, Sep. 2017.

[36] M. Harman, "Search based software testing for android," in *Proc. IEEE/ACM 10th Int. Workshop Search-Based Softw. Test. (SBST)*, May 2017, p. 2.

[37] J. Guiochet, M. Machin, and H. Waeselynck, "Safety-critical advanced robots: A survey," *Robot. Auton. Syst.*, vol. 94, pp. 43–52, Aug. 2017.

[38] C. Yu and J. Xi, "Simultaneous and on-line calibration of a robot-based inspecting system," *Robot. Comput.-Integr. Manuf.*, vol. 49, pp. 349–360, Feb. 2018.

[39] M. Jasiński, J. Mączak, P. Szulim, and S. Radkowski, "Autonomous agricultural robotp—Testing of the vision system for plants/weed classification," in *Proc. Conf. Autom. (AUTOMATION)*, vol. 743. 2018, pp. 473–482.

[40] J. Brookes *et al.*, "Robots testing robots: ALAN-Arm, a humanoid arm for the testing of robotic rehabilitation systems," in *Proc. Int. Conf. Rehabil. Robot. (ICORR)*, Jul. 2017, pp. 676–681.

DEBDEEP BANERJEE received the master's degree in electrical engineering from the Illinois Institute of Technology. He has over 10 years of industry experience in the field of software/systems engineering. He is the Software/Systems Development Engineer and the Test Lead for the computer vision project. He is currently a Senior Staff Engineer and an Engineering Manager with the Qualcomm Technologies, Inc., USA. He is responsible for the test automation design, planning, development, deployment, code reviews, and project management. He closely involves with software/system teams and gathers test requirements for the project. He has been involving with the software test automation team, since the inception of the computer vision project in Qualcomm Technologies Inc. He is also involved in managing and developing software for the Computer Vision Laboratory using the robotic arm.

KEVIN YU is currently a Test Engineer with the Qualcomm Technologies, Inc., USA, and has contributed to test automation validation for continuous integration for validating 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 the Qualcomm Technologies, Inc., USA, and has actively involved on MATLAB post-processing modules for CV features and various other automation projects.

• • •