IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 13, 2019, accepted January 5, 2020, date of publication January 13, 2020, date of current version January 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2965947

Object Removal Software Test Automation

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 This paper considers the problem of designing a test method and robust test automation for evaluating the functionality of computer vision algorithms for object removal and image reconstruction of the removed area. An object removal algorithm is used to select and remove a region in an image. After the removal of the object, the object removal algorithm regenerates the area in the image from which the object was removed, similar to photoshop. We developed an algorithm in MATLAB that accesses the features of the macroblocks from the region that adjoins the removed object and correlates these macroblocks with the actual area that is regenerated. The SSIM (structural similarity index) score is calculated by comparing the ground truth with the image generated from image reconstruction after removal of the object. Finally, we calculate the artifact area present in the regenerated region to determine the algorithm quality. To validate this method, the authors compare a 3rd-party object removal solution with an in-house developed solution using the proposed approach. This comparison shows that the SSIM score is 1.55% higher than the 3rd-party solution and 26.3% lower in artifact areas. Besides the results, this paper also describes in detail the procedure to automate the use case of object removal test automation to scale up the test coverage.

INDEX TERMS Software engineering, software testing, image processing, image restoration, multimedia.

I. INTRODUCTION

This paper discusses a validation strategy for testing an object removal algorithm. We designed test-automation-based algorithms that evaluate the area that is regenerated after an object is removed. The object removal software test automation is developed to validate the functionality, performance, and stability of the object removal algorithm. The test automation uses image quality algorithms such as the SSIM (structural similarity index) to evaluate the comparison of the image that was reconstructed after the object was removed with the ground truth. We also use algorithms to detect the presence of artifacts in the image reconstructed area.

We compare the luma values of the macroblocks of the region that adjoins that from which the object was removed. Then, we apply the k-means clustering algorithms to confirm the number of color clusters in the images and compare this number with the ground truth to evaluate the functional verification of the image reconstruction.

There has been research in the field of denoising and artifact removal of an image using Weiner filter etc., [1]. The effect of object removal from an image has to be filled in a visually plausible way [2]. We develop a test automation that measures the quality of the regenerated area using MATLAB software after the object removal algorithms are used. The challenge for the validation team is to programmatically determine the quality of the removed objects from the images. The object removal algorithm needs to regenerate the removed area so that it blends with the surrounding pixels of the object. We identify an approach to effectively generate test images and ground truth images. The output image results can be compared to the ground truth image using the SSIM (structure similarity index) measurement to quantify the regenerated area quality. Finally, a computation is performed using machine learning algorithms such as k-means clustering to detect artifacts and obtain the artifact area percentage. We deterministically assign a result to these test cases. Saliency maps are applied for determining objects and applying masks (green in color) to them based on the object detection.

The object removal applications require us to select a region in the object to remove. If we apply a mask to the object that needs to be removed, we can apply swipe events for the area selected in the mask through touch event-based automation. The challenge is to detect and select the object and then color the object for the mask. We apply saliency maps for object detection using MATLAB, and we color the region of the saliency map for the mask generation.

The results of a comparison with other Android applications performing object removal are obtained, and testing is

The associate editor coordinating the review of this manuscript and approving it for publication was Habib Ullah.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

performed with different test vectors, including a cluttered background, a non-cluttered background, etc. We perform competitive analysis tests with a 3rd-party software application using the test automation on different types of background test images, quantitatively proving that our object removal application regenerates background areas that have better image quality. We compare the object removal latencies for the device that performs with hardware processors and that uses hardware acceleration for performing the object removal algorithms. For the results, the comparative Android application does not use hardware acceleration on the hardware processors. We observe noticeable gains in the latency measurements for the object removal algorithms with hardware acceleration.

This test automation strategy is evaluated for inclusion at various software integration points so that we can detect any regressions in the software code churns. If a regression is detected, the failures re triaged, and if a failure is diagnosed as a software issue, it is debugged and fixed by the software development team. This approach helped us maintain the software quality and detect issues early in the software development lifecycle. In particular, observing problems late in the software development process can lead to significantly higher costs of solving the problem.

II. MOTIVATION

The goal of developing the test automation strategy was to measure the accuracy and performance of object removal algorithms qualitatively. The development of a robust automation helps us consistently benchmark the functionality and performance test results.

The test automation strategy enables us to apply machine learning algorithms such as k-means clustering to perform color segmentation of the image. The test automation strategy performs feature extraction of the adjoining macroblocks from which the object was removed and compares them to the regenerated area. Depending on the correlation values, the test automation strategy can objectively assess a pass or fail result for the test case. The test automation strategy also uses correlations to compare the similarity of the features of the adjoining macroblocks to the regenerated area. Any unusual colors that are formed during the regeneration of the removed object can lead to the creation of additional color segments and decrease the correlation values. Therefore, the test automation strategy can detect abnormal color patches, which are distortions in the regenerated area from which the object has been removed.

Stability tests are crucial to software testing but executing the same test cases multiple times can be tedious. Thus, writing and deploying a reliable test automation strategy that can perform functional validation of the test cases can be beneficial. In addition, many issues, such as memory leaks, are detected after several iterations of testing. These stability issues may not be detected during functional testing. The test automaton collects logs for each iteration, which helps the engineers effectively triage the failures based on log analysis and the captured images.

We intend to reduce the workload of manual testing. Manual testing can be a tedious and error-prone process. Thus, if the automation is reliable, test engineers can focus on new test software development efforts, writing new test plans and enhancing test coverage by carefully evaluating the software architecture of the algorithms and developing new test automation strategies for testing them. The test automation strategy can be deployed at several points of the software integration cycle. Testing code churn early in the software development process can lead to the detection of regression issues. The test case failures are triaged and then software failures are debugged and fixed by the software developers. Using this approach, we can efficiently detect the issues early in the process.

III. BACKGROUND AND RELATED WORK

The research field of artifact removal of an image involves development of image reconstruction techniques. Image reconstruction requires image smoothing techniques. The image smoothening algorithms use techniques for reconstructing the smoothly changed shape and intensity of the original image [3]. Research has been done to apply deep network architecture for removing rain streaks from an image. This has helped in improved rain removal from images and much faster performance in object removal techniques [4]. The image reconstruction techniques are discussed after either part of the image is corrupted [5].

A clustered differential pulse code modulation method with removal of local spectral outliers has been studied and analyzed for lossless compression of images [6]. Image processing algorithms like 3D reconstruction and hand jitter reductions has been implemented on mobile phones. Validation techniques involve design and development of test automation for validating these image processing algorithms [7], [8]. Techniques have been developed to detect the types of noise artifacts for removal in experimental data and analysis [9]. Test automation development for embedded software-based image rectification techniques has been done for validating the automatic color, sharpness of images [10]. Face recognition software has been popular for image processing in mobile phone. Validation techniques for this feature testing of face recognition has been discussed with performance metrics [11].

Test automation exists for automation image processing systems of high precision X Ray and Neutron imaging applications [12]. Research has been done for development of testing methods for designing artificial scenes and automatically generating virtual images with precise annotation [13]. Test systems can calculate the accuracy of image processing capture systems like fisheye cameras [14]. The accuracy calculations are crucial for any test system to analyze the performance of the system and the software. In this paper we discuss the test automation strategies for the object removal algorithm.

TABLE 1. Snapshot of test cases in the object removal test plan.

Test Name	Background	Object to Remove
Remove multi-colored object from uniform background	White wall	Mannequin head
Remove uniform simple shaped object from textured background	Textured fabric	Sticky note
Remove complex colored object from repeated pattern background	Brick wall	Wall Poster
Remove complex shaped object from complex background	Wall poster	Building model

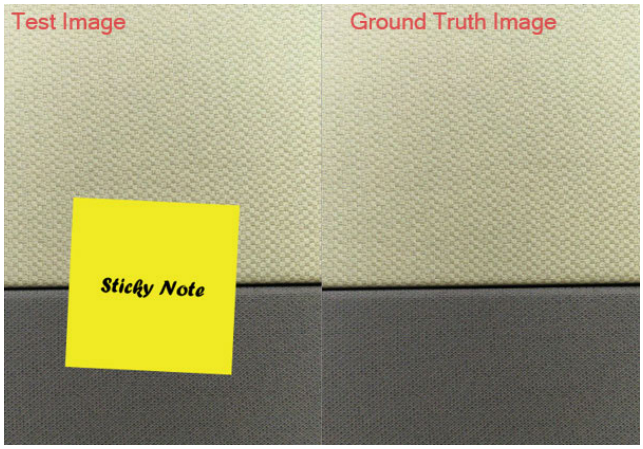


FIGURE 1. Regenerated test image and ground truth image pair.

IV. SOLUTION

A. SOFTWARE OVERVIEW

Object removal is a digital image editing solution. By editing an image, the user can remove the unwanted content from the image. The libraries provide an application programming interface (API) in which the masked region is an input parameter, providing flexibility to the user to select the content of his/her choice. The object removal feature is preferred when obtaining more personalized images.

B. OBJECTIVES

The goal of the object removal software is to remove the content in the selected regions and fill-in the holes of these regions in a visually plausible way. An end-to-end test application automation framework is required to test these features for a large data set and quantitively evaluate the object removal quality.

In our object removal test plan, there are test cases that involve with combination of different types of background and objects to remove. The backgrounds can be categorized into uniform colored simple background, textured background, repeated pattern background, and complex color/textured background. The objects to be removed from the backgrounds are uniform and multi-colored objects as well as simple and complex shaped objects. The combination of these objects and backgrounds form a good representation of scenarios that user will be using the object removal app for. Table 1 shows a snapshot of sample test cases in our object removal test plan.

Since we have hundreds of these test cases to fully evaluate the object removal software's performance, manually testing them is time consuming and object removal quality result is subjective. An automated process is needed for object removal software testing. The main challenges of automating this process are the remove object selection, ground truth/test image generation and object removal/background restoration quality postprocessing. Our objectives are to find automated solutions to these pain points and to establish a complete automated workflow.

C. GROUND TRUTH IMAGE AND TEST IMAGE GENERATION

To effectively evaluate the object removal and background restoration quality, a perfect background is needed as the ground truth before applying object removal. And we have come up with two methods to generate ground truth image and test image.

One method is to select available images that are only composed of background that are either procured from gallery or image database and use them as ground truth images. Then, we apply image overlap using image cloning software such as Photoshop, and subsequently add object images on top of the background image. The test images are generated composed with the object and the background, as shown in figure 1.

The advantage of this method is the ground truth image is the same as the background in the test image. The image comparison result in the post-processing stage will be highly accurate. But the problems for this method are the object image has to be edited using image software cut to its own shape before overlap to the background, and also the composed image does not fully represent the natural scene taken with both foreground object and background in the same picture.

The other method of generating ground truth and test images is by taking snapshot of the actual background and object placed in front of the background using camera. When large number of real-world test data images are needed, a systematic way of generating them is necessary. We have utilized the latest technology of robotic arm system to help capture these ground truth and test images.

We have a 6-dof motion robotic arm setup in a closed-space test lab, where we have full control of light intensity and background setup, as shown in figure 2. Different backgrounds can be setup on multiple regions of the lab wall. They range from single color to multiple colors, single texture to complex texture, representing the background requirements of the test cases. The objects from single to complex shapes in the test cases requirements are placed in front covering part of

FIGURE 2. Robotic arm pointing the camera at different test scenes.

FIGURE 3. Ground truth and test images captured using the robotic arm.

the backgrounds. The combination of the objects and the backgrounds create the scenes for the ground truth and test images. The challenge is to capture these images properly so that the background in the ground truth image appears the same in the composition test image for accurate comparison.

Using a high precision robotic arm, there will be only ±0.03 mm position error for every meter the robot moves. This means the robotic arm is able to hold the camera device and move it to the accurate position to take snapshots every time. A 6-dof robotic arm has 6 joints, by rotating these joints using controller the tip of the robotic arm can be extended to the desired position to take snapshots. The robot position can be stored to the controller and use robot programming software like WINCAPS3 or ORiN2 to automate the robot motions moving through each saved position in front of different backgrounds. Using robotic arm for ground truth and test image capture proves to be much more accurate than manual capture by analyzing the comparison results. And it also saves valuable time in this test data generating process. On the camera side, the settings like focus and exposure have to be set off from automatic mode. This is needed to prevent exposure and sharpness difference between the ground truth and the test images. The captured results are shown in figure 3.

Generating ground truth and test images using image capture method has its advantages. The generated images have more realistic compositions like the shadows and light exposures. Also, when the required background and test images are not available, we can use this method to create the scenes and generate the test data. The downside for image capture method is there will always be a delta between the ground truth image background and the test image background. However, this delta can be minimized by using the robotic arm for image capture and have the right camera settings. Depending on the test case requirements, the image capture method and the image overlap method can both be effective in generating ground truth and test images.

D. GENERATION OF THE GREEN MASK BY OBJECT DETECTION

The next test automation challenge is the removal of the region selection. Using fingers or a pen to highlight the removal region can be difficult on complex shapes. In addition, this approach has a large capacity for human error over repeated attempts. We identify an approach to select the object to be removed in an image by highlighting it in a green mask. We apply saliency maps [15]–[17] to detect objects in the images. The saliency maps are generated based on the features of the images, such as color, intensity, etc. The saliency maps are normalized and overlaid on the original image for object detection.

The saliency maps can be created based on the graph-based saliency algorithm. If we are given an image I the most significant location of the images can be computed. The feature maps based on color, intensity etc are computed and then these feature maps are normalized for generate the saliency map. M(i, j) corresponds to locations in the image I which are unusual to its neighborhood and can be related to higher values of the activation map.

To calculate the dissimilarity between the maps of M(i, j) and M(p, q) the following formula is used.

$$d\left((i,j) \parallel (p,q)\right) \triangleq \left|\log \frac{M\left(i,j\right)}{M\left(p,q\right)}\right|$$

Once the saliency map is generated a weight will be associated to the directed edge from node (i, j) to node (p, q).

$$w_1\left((i,j),(p,q)\right) \triangleq d\left((i,j) \parallel (p,q)\right) \cdot F\left(i-p, j-q\right),$$

$$\text{where } F\left(a,b\right) \exp \triangleq \left(-\frac{a^2+b^2}{2\sigma^2}\right).$$

Here sigma is a free parameter of our algorithm. Once the feature maps are formed comprising of individual features like color, intensity etc the feature maps are normalized. Winner Takes All Computation to detect the point of highest salience in the map at any given time. This information is used for compute object detection and recognition.

Once the saliency maps are generated and the objects are detected, we use the map area to color the object and generate the mask. Please refer to figure 4a, 4b and 4c for details of the saliency map generation and the generation of the masks in the image.

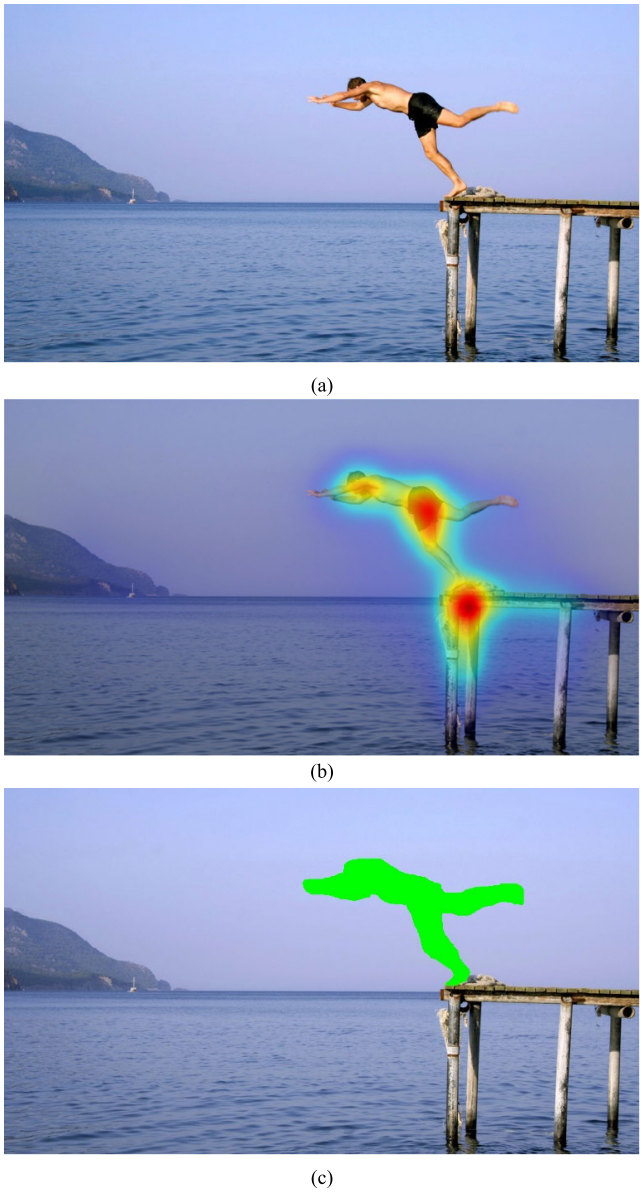


FIGURE 4. (a) The original image from which the object (man diving into the water) needs to be removed. (b) The saliency maps are generated and overlaid on the image. Please note that the saliency maps are used for object detection. We use MATLAB code for the saliency map generation. (c) The saliency maps are generated, and the area of the saliency maps is used for coloring the area and generating the mask.

We then use these masks in our test automation. The green mask is selected using touch events, and the object removal algorithms are applied to remove the area in the mask. The object removal algorithm then regenerated the removed area. The algorithm analyzes the features, such as color and luma values of the adjoining pixels of the removed area, and then regenerates the area.

E. OBJECT REMOVAL AND BACKGROUND RESTORATION POSTPROCESSING ALGORITHM

In this paper, we propose a postprocessing algorithm for analyzing the quality of a reconstructed region. The algorithm

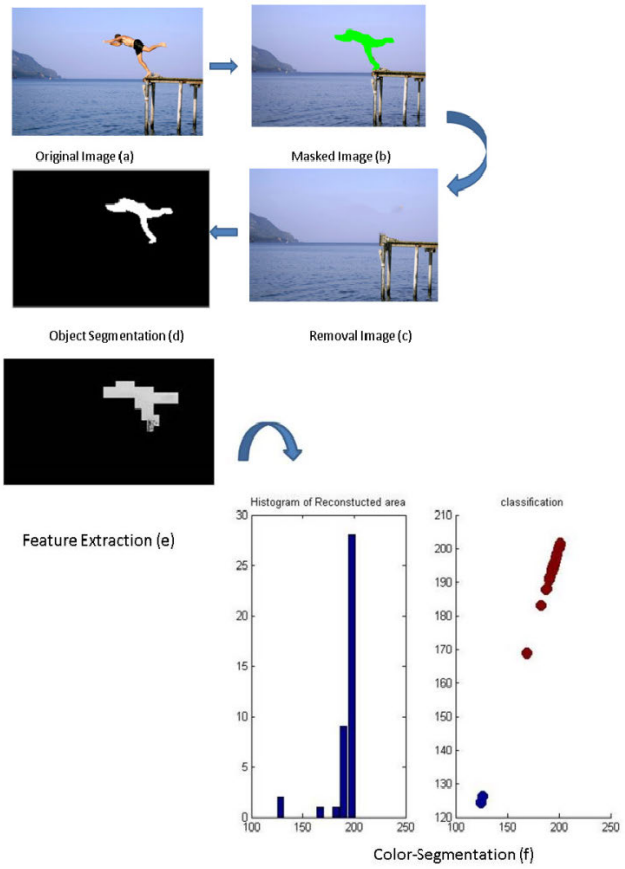


FIGURE 5. Stepwise presentation of the algorithm: (a) Original image with the object; (b) Masked object with green color; (c) Resultant image after using the object removal feature; (d) Object segmentation from the masked image; (e) Feature extraction from the removal image; (f) Color segmentation using the luminance value.

is designed to segment the object from the image and find the neighboring pixel blocks using the standard deviation. Then, the luminance value is calculated for feature extraction. Finally, a computation is performed using SSIM (structural similarity index) and k-means clustering to obtain the removal artifact percentage.

1) OBJECT SEGMENTATION

Object segmentation is required to identify the erased object in the image. The object is defined by using a predefined mask with $RGB = (0, 255, 0)$ values. In this manner, the object is segmented in the form of a binary image, which is denoted as $I(x, y)$ and satisfied $I(x, y) = I(x, y, 1) == 0, I(x, y, 2) == 1, I(x, y, 3) == 0$.

2) FEATURE EXTRACTION

The information of the neighboring pixels is needed to extract the information of the filled area in the output image. The neighboring pixels of the object are identified by splitting the binary image into overlapping blocks of $b \times b(32 \times 32)$ pixels. In this way, the whole image of size $M \times N$ is divided into $(M - b + 1) \times (N - b + 1)$ macroblocks. To identify

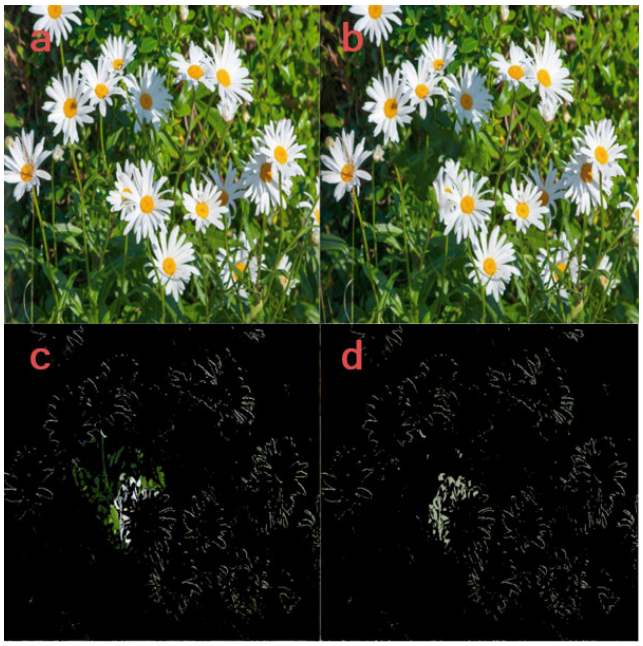


FIGURE 6. Object removal/reconstruction artifact extraction: (a) Ground truth background image; (b) Reconstructed background image after object removal; (c) Color segmentation comparison between the output and ground truth images; (d) Single color artifact extraction.

the blocks of interest, the standard deviation is calculated for each macroblock. The maximum deviation is observed in the blocks that contain the object pixels and neighboring pixels. For feature extraction, the output image is also divided into overlapping blocks of $b \times b(32 \times 32)$ pixels. The blocks that had maximum deviation and contained the removal object are selected for feature selection for further processing. Feature selection is performed by calculating the luminance value ($Y = 0.299R + 0.587G + .114B$) of each selected block. Luminance quantifies the trichromatic intensity perception of humans [4].

3) COMPUTATION

After identifying the blocks that contain the reconstructed region and their neighbors in the output image, a computation is performed using SSIM (structural similarity index) between the output image and the ground truth image on the reconstructed region. A high SSIM score (close to 1) means the object removal has performed well, and the reconstructed background that is generated is almost identical to the original ground truth image. For artifact detection and calculation on the output image, color segmentation is performed on the output image and compared to the ground truth image using the k-means clustering algorithm. This approach highlights the areas where there are color differences between the two images. Then, the delta regions are extracted into a single color, highlighting the artifacts. This process is demonstrated in figure 6.

F. TEST AUTOMATION WORKFLOW

With the solutions to all three pain points, a complete test automation workflow can be established. Before running the automation, background images are set as ground truth images. The background images are then added with objects layers on top. Using object detection, these objects are predefined with green masks as the input images. When the object removal test automation is executed on the input image, a Perl script is written to open the Android test application, to select the predefined mask using touch events, to perform the removal, and to save the edited image. The postprocessing algorithm evaluates the quality of the regenerated area based on the features of the macroblocks that are near the object that is being removed and then applies the SSIM (structural similarity index) scores. The postprocessing algorithm also applies the k-means clustering algorithm to the image to perform color clustering to identify artifacts. A computation is performed using k-means clustering to obtain the artifact percentage. Please refer to figure 7 for details.

TABLE 2. Post processed image result scores comparison table.

Types of background	Number of images tested	SSIM Score		Artifact Percentage	
		Object removal app	3rd-party app	Object removal app	3rd-party app
Uniform background	45	0.971	0.968	0.37%	0.40%
Textured background	62	0.872	0.839	4.39%	6.64%
Repeated background	38	0.684	0.679	11.94%	13.23%
Complex/ cluttered background	55	0.843	0.839	3.94%	6.71%

V. RESULTS

A. OBJECT REMOVAL PERFORMANCE RESULTS ON IMAGE QUALITY

In our test automation, we test 200 test vectors, including the dataset images from Caltech101 [18]. The images are categorized with different backgrounds such as uniform, textured, repeated and cluttered. These different types of backgrounds represent different challenges for the background restoration algorithms. To compare the performance with other object removal apps on the market, we perform the end-to-end automated test on both the object removal app and the 3rd-party app. The postprocessed test scores are averaged and are shown in table 2.

From this table, we can see that the object removal app and the 3rd-party app performed similarly on the same type of backgrounds, while the object removal app received slightly better SSIM scores (approximately 1.55%) on all four types of backgrounds. The SSIM scores comparison data are plotted as bar graphs in figure 8.

The SSIM score measures how similar the comparison images are; in this case, the images are the output image and the ground truth image. These data show that the SSIM

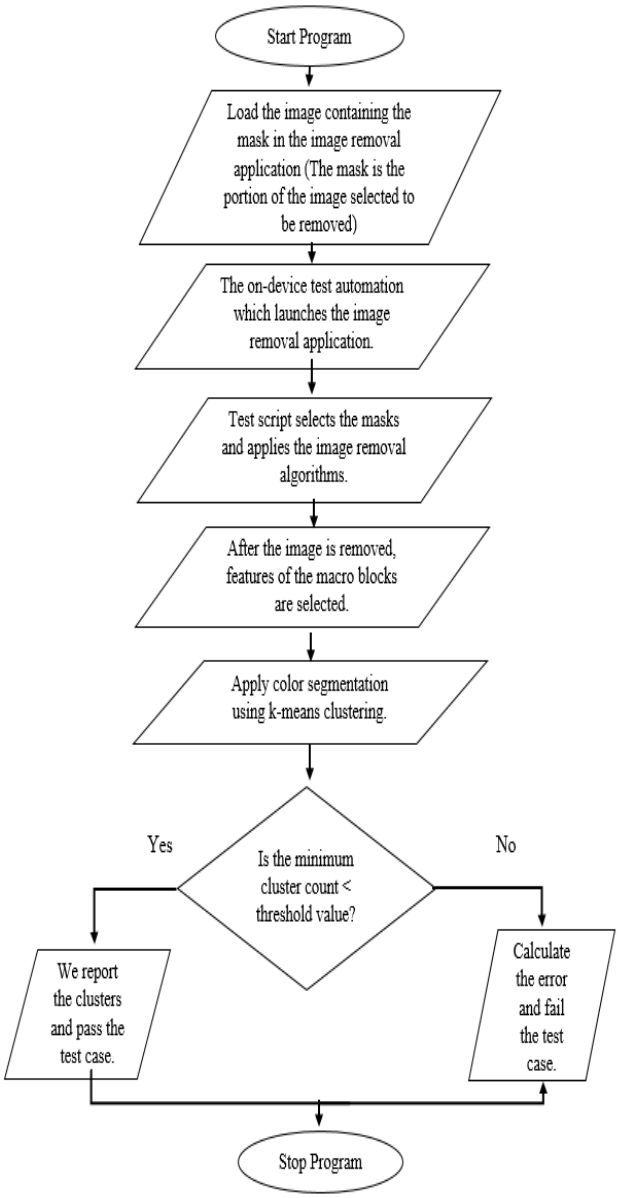


FIGURE 7. Flowchart of the end-to-end object removal test automation.

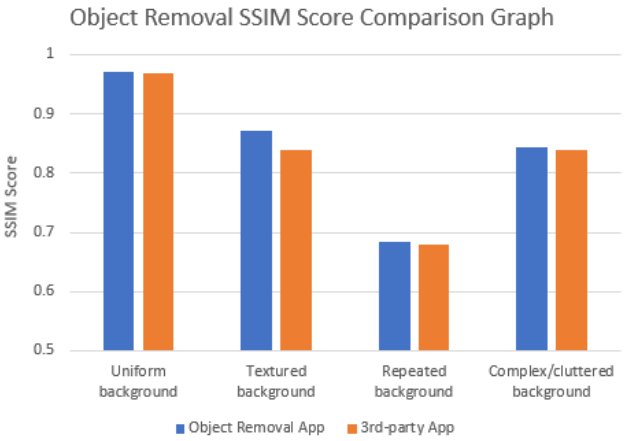


FIGURE 8. Comparison of object removal SSIM scores between the object removal app and the 3rd-party app.

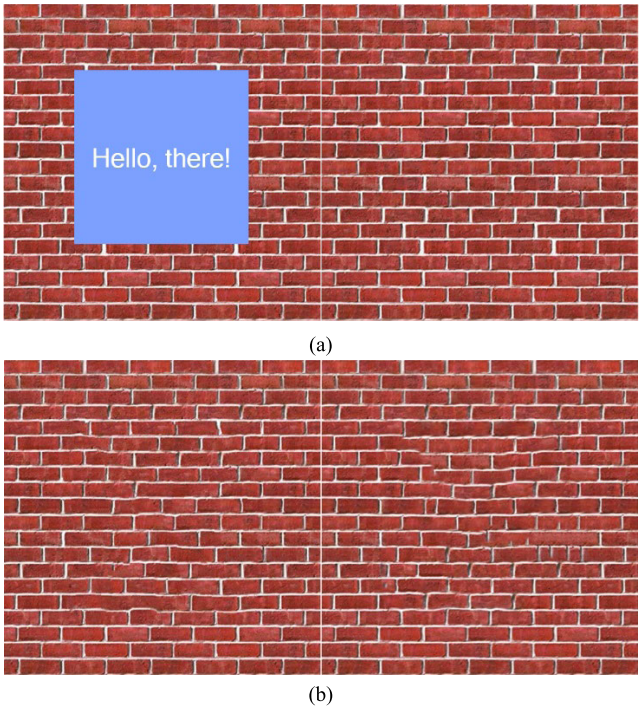


FIGURE 9. (a) Image of a brick wall with a blue poster and ground truth image without the poster. (b) Object removal output images (left one from the object removal app and right one from the 3rd-party app).

scores vary between different types of backgrounds. For uniform colored backgrounds, it is easier to reconstruct the background of the removed region. Therefore, this category received the highest SSIM score. On the repeated background images, the background contains even or uneven repeated patterns. The background reconstruction algorithm must analyze these patterns and fill the removed space with the proper patterns. This task proves to be quite challenging, and this category received the lowest SSIM score. As an example, figure 9a shows the image of a brick wall before the removal of the blue poster region and the ground truth image of the brick wall without the poster. In figure 9b, the left side is the object removal app output of the reconstructed brick wall image after poster removal, and the right side is the 3rd-party output image. It is clear that both software solutions were unable to reconstruct the wall pattern perfectly compared to the ground truth. However, the left image's reconstructed pattern more closely resembles the actual pattern.

Another example of the textured background object removal output results is shown in figure 10. Both object removal software solutions remove the yellow sticky note on the image from figure 1.

Both output images are able to reconstruct the removed sticky note area with the proper surrounding texture. However, both images have an artifact of a skewed line on the space separating the two textures. We also occasionally observe that when selecting the removal regions with the 3rd-party app, a diagonal line is applied to the image

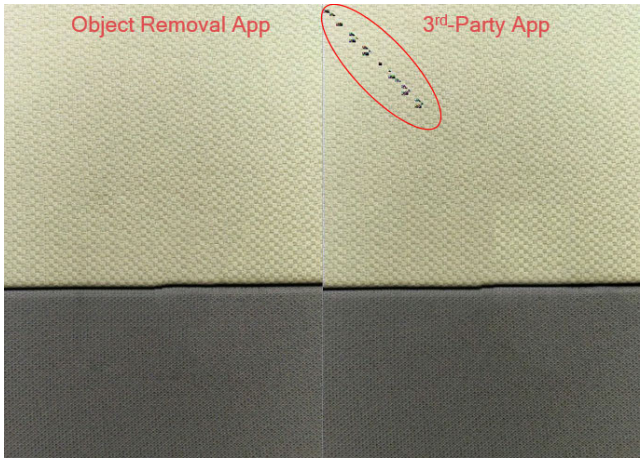


FIGURE 10. Textured background object removal output images from the object removal app and the 3rd-party app.

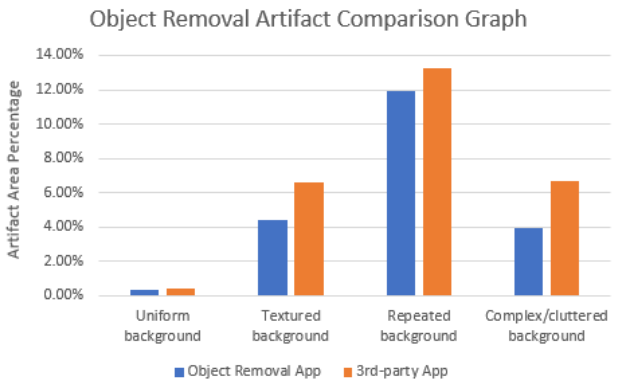


FIGURE 11. Object removal test output artifact area percentage comparison graph.

along with the removal region. This line creates an additional artifact on the output result image. This issue emphasizes the importance of artifact detection and measurement in the object removal test automation.

The artifact percentage used in our test automation is calculated by dividing the areas of the processed output image highlighted with the artifact by the area of the removed (masked) regions. The artifact percentage data from table 1 is converted into a bar graph in figure 11. This graph shows that the object removal app generally has smaller artifact regions (approximately 26.3%) than the 3rd-party app. The uniform background category shows almost zero artifacts. Textured and complex/cluttered backgrounds have a moderate region of artifacts. In addition, repeated backgrounds have the largest region of artifacts, as expected. This artifact result is aligned with the SSIM score result.

Figure 12 shows an example of the processed output images for artifact detection. These images are created by removing a butterfly from the flowers previously used in figure 6. The left image in figure 12 is processed from the object removal app, and the right image is processed from the 3rd-party app. The highlighted regions have artifacts. The image on the right clearly shows a larger highlighted area.

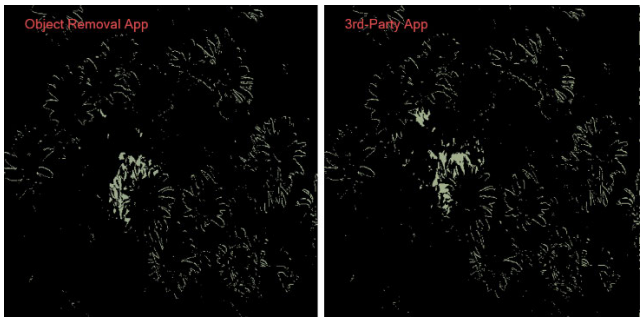


FIGURE 12. Processed artifact areas from the object removal app output and the 3rd-party app output.

TABLE 3. Object removal latency result comparisons.

Types of background	Image resolution	Object Removal App latency (sec)	3rd-Party App latency (sec)
Uniform	1280x720	0.16	0.90
Textured	1280x720	0.53	1.10
Repeated	1280x720	0.39	0.96
Cluttered	1280x720	0.84	1.41

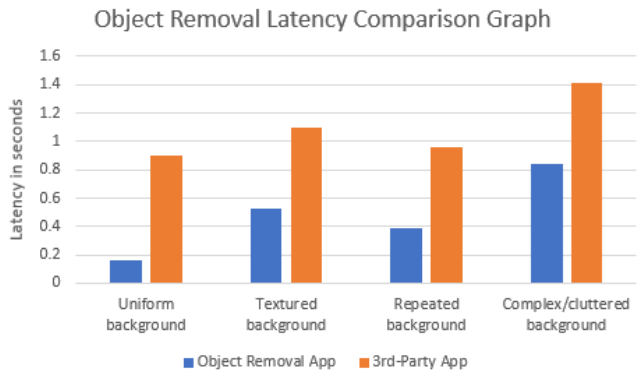


FIGURE 13. Latency comparison graph on object removal process.

From running the automated test on both object removal software apps with the 200-test image dataset, we can conclude the object removal app performs better than the 3rd-party app in object removal quality. In addition, we find that the algorithm was most accurate for images with the following characteristics:

1) Constant color background. (e.g., clear sky)
2) Small textures.
3) Sharp edges.
4) Small mask sizes.

B. OBJECT REMOVAL PERFORMANCE RESULTS ON LATENCY

During the object removal test, we also measure the latency of the object removal process on the hardware accelerated object removal app and the 3rd-party Android app. The measurement is taken from Android logs, and it is measured across all four

categories of backgrounds. The latency results are averaged and are presented in table 3.

The latency comparison data are converted to bar graphs shown in figure 13 for a better visual comparison.

From the results, we concluded that the latency for the object removal is approximately 56% lower in the case of the object removal app that use hardware acceleration than that of the 3rd-party Android app. The hardware acceleration is achieved by using signal processors, which executes the object removal algorithms. The complexity of the object removal algorithms increases with cluttered backgrounds, and we observe that the latency increases as we test from uniform backgrounds to cluttered backgrounds.

VI. CONCLUSION

The object removal test method and automation strategy provide an option for executing algorithm tests for each software release and benchmarking the results by the software release. New test contents can be added to the test automation framework to expand the test coverage for more specific scenario object removal algorithm testing.

The proposed test method and automation strategy can be used for executing functional, performance, stability and adversarial test cases. They can also be scaled to multiple software products and produces consistent and reliable test results. We are constantly reviewing the test plan and adding new test cases according to the new features and optimizations that are added by the software development teams based on the object removal feature. The goal is to scale up the test automation so that test engineers can utilize their time for the development of new feature tests and to ensure that this test automation strategy can execute regression test cases reliably.

REFERENCES

[1] A. Kazakeviciute, C. J. H. Ho, and M. Olivo, "Multispectral photoacoustic imaging artifact removal and denoising using time series model-based spectral noise estimation," *IEEE Trans. Med. Imag.*, vol. 35, no. 9, pp. 2151–2163, Sep. 2016.

[2] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. Image Process.*, Vol. 13, no. 9, pp. 1200–1212, Sep. 2004.

[3] C. Cho and S. Lee, "Effective five directional partial derivatives-based image smoothing and a parallel structure design," *IEEE Trans. Image Process.*, vol. 25, no. 4, pp. 1617–1625, Apr. 2016.

[4] X. Fu, J. Huang, X. Ding, Y. Liao, and J. Paisley, "Clearing the skies: A deep network architecture for single-image rain removal," *IEEE Trans. Image Process.*, vol. 26, no. 6, pp. 2944–2956, Jun. 2017.

[5] S. D. Rane, G. Sapiro, M. Bertalmio, "Structure and texture filling-in of missing image blocks in wireless transmission and compression applications," *IEEE Trans. Image Process.*, vol. 12, no. 3, pp. 296–303, Mar. 2003.

[6] J. Wu, W. Kong, J. Mielikainen, and B. Huang, "Lossless compression of hyperspectral imagery via clustered differential pulse code modulation with removal of local spectral outliers," *IEEE Signal Process. Lett.*, vol. 22, no. 12, pp. 2194–2198, Dec. 2015.

[7] D. Banerjee, K. Yu, and G. Aggarwal, "Robotic arm based 3D reconstruction test automation," *IEEE Access*, vol. 6, pp. 7206–7213, 2018.

[8] D. Banerjee, K. Yu, and G. Aggarwal, "Hand jitter reduction algorithm software test automation using robotic Arm," *IEEE Access*, vol. 6, pp. 23582–23590, 2018, doi: 10.1109/ACCESS.2018.2829466.

[9] D. Allman, A. Reiter, and M. A. L. Bell, "Photoacoustic source detection and reflection artifact removal enabled by deep learning," *IEEE Trans. Med. Imag.*, vol. 37, no. 6, pp. 1464–1477, Jun. 2018.

[10] D. Banerjee, K. Yu, and G. Aggarwal, "Image rectification software test automation using a robotic Arm," *IEEE Access*, vol. 6, pp. 34075–34085, 2018.

[11] D. Banerjee and K. Yu, "Robotic Arm-based face recognition software test automation," *IEEE Access*, vol. 6, pp. 37858–37868, 2018.

[12] J. A. Hashem, M. Pryor, S. Landsberger, J. Hunter, and D. R. Janecky, "Automating High-precision X-ray and neutron imaging applications with robotics," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 663–674, Apr. 2018.

[13] Y. Tian, X. Li, K. Wang, and F.-Y. Wang, "Training and testing object detectors with virtual images," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 2, pp. 539–546, Mar. 2018.

[14] J. Schneider, C. Stachniss, and W. Forstner, "On the accuracy of dense fisheye stereo," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 227–234, Jan. 2016.

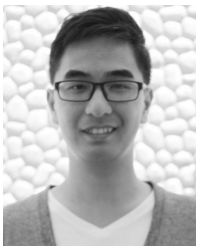
[15] J. Harel, C. Koch, and P. Perona, "Graph-Based Visual Saliency," in *Proc. Neural Inf. Process. Syst. (NIPS)*, 2006.

[16] J. Harel. *A Saliency Implementation in MATLAB*. Accessed: Oct. 10, 2019. [Online]. Available: http://www.klab.caltech.edu/~harel/share/gbvs.php

[17] X. Hou, J. Harel, and C. Koch, "Image signature: Highlighting sparse salient regions," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 194–201, Jan. 2012.

[18] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshop*, Jun./Jul. 2004.

DEBDEEP BANERJEE (Member, IEEE) received the master's degree in electrical engineering from the Illinois Institute of Technology. He has more than ten years of industry experience in the field of software/systems engineering. He is currently a Senior Staff Engineer and the Engineering Manager with Qualcomm Technologies, Inc., USA. He is also the Software/Systems Development Engineer Test Lead of the computer vision project and is responsible for test automation design, planning, development, deployment, code reviews, and project management. He also works closely with the software/system teams. He has been working with the Software Test Automation Team since the inception of the computer vision project at Qualcomm and also involved in managing and developing the robotic arm software used in the Computer Vision Laboratory.



KEVIN YU has contributed to test automation validation for continuous integration and regression tests for computer vision algorithms. He has also validated computer vision engine features such as image rectification for Android software products. He is currently a Senior Test Engineer with Qualcomm Technologies, Inc., USA.

GARIMA AGGARWAL has actively worked on MATLAB postprocessing modules for CV features and various other automation projects. She is currently a Senior Test Engineer with Qualcomm Technologies, Inc., USA.