Exploration of 3D Texture and Projection for New CAPTCHA Design

Simon S. Woo[1,2(✉)], Jingul Kim[1,3], Duoduo Yu[1], and Beomjun Kim[1]

[1] Computer Science Department, University of Southern California, Los Angeles, CA, USA
{simonwoo,jingulki,duoduoyu,beomjun}@usc.edu

[2] Information Sciences Institute, Marina Del Rey, CA, USA
simonwoo@isi.edu

[3] Korea Army Academy at Yeongcheon, Yeongcheon, Republic of Korea

Abstract. Most of current text-based CAPTCHAs have been shown to be easily breakable. In this work, we present two novel 3D CAPTCHA designs, which are more secure than current 2D text CAPTCHAs, against automated attacks. Our approach is to display CAPTCHA characters onto 3D objects to improve security. We exploit difficulty for machines in rotating 3D objects to find a correct view point and in further recognizing characters in 3D, both tasks that humans can easily perform. Using an offline automated computer vision attack, we found that 82% of the new text reCAPTCHA characters were successfully detected, while approximately 60% of our 3D CAPTCHAs were detected only if characters were focused and zoomed from the direct view point. When CAPTCHAs are presented in slightly different views, the attack success rates against our approaches are reduced to almost 0%.

Keywords: CAPTCHA · Authentication · 3D

1 Introduction

The Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) [27] is a type of challenge and response test to distinguish humans from machines. Although CAPTCHAs must be easy for humans to solve, they must be difficult for a bot to pass in a reasonable amount of time. Currently, the most widely used form of a CAPTCHA, 2D text CAPTCHA, presents a CAPTCHA as a combination of random characters, symbols, and numbers in a 2D space. Due to its simplicity and ease of use, 2D text CAPTCHAs have been widely deployed in many websites to prevent access from automated bots. However, 2D text CAPTCHA is vulnerable and even easily breakable by Optical Character Recognition (OCR) software or automated offline post-processing attacks such as segmentation attacks [8,20] that segment and recognize each character separately. There have been attempts to add more background noise, dots, and lines across characters to make recognition difficult. However, automated character recognition technology is getting better every day. Therefore, there is a fundamental limitation to 2D text CAPTCHAs, which can be easily broken by improved OCR applications and segmentation attacks.

In order to overcome the fundamental limitations of 2D text CAPTCHA, audio based CAPTCHAs [2] have been proposed. However, [18] shows that an automatic speech recognition (ASR) system can defeat audio reCAPTCHA with a significantly high probability of success. Also, moving-image object recognition (or video) based CAPTCHAs [12,22,24] have been proposed, where characters are moving or streaming as a video and users enter animating 2D characters. However, [28] provides an effective attack against these types of CAPTCHAs.

In this work, we address the weakness of the current text CAPTCHA mechanisms from a fundamentally different direction. We either place or project 2D text CAPTCHAs on the surface of a 3D object, and then randomize the viewpoint of the 3D object. Thus, users need to rotate objects first to get the right view points, and the identify characters. We explore two different approaches to place 2D text CAPTCHA onto 3D objects. The first approach is to *embed* CAPTCHA characters as a part of textures in a 3D object using UV mapping [5]. The second approach is to *project* a 2D text CAPTCHA onto a 3D object. The first approach explores various texture rendering techniques on the surface of a 3D object to improve security and usability. The second approach projects an individual CAPTCHA character from a specific view point towards uneven surfaces of a 3D object using ray-tracing.

While humans can easily perform the object rotation and character recognition task in 3D, bots require additional complex algorithms to identify the correct shape of a 3D object and pin-point the coordinates that CAPTCHA characters are placed on. Then, bots have to perform character recognition and segmentation to recognize characters displayed on the 3D objects. Although significant progress has been made in 3D shape identification and character recognition research [7,11,16,26], we demonstrate that a series of object recognition, rotation, and CAPTCHA recognition tasks in 3D are still challenging given the current bots' capabilities.

2 Related Work

Many types of CAPTCHAs have been proposed using video, audio, game, puzzle, cognitive task, etc. Due to space limitations, we mainly surveyed research that is directly relevant to our approaches.

Text-based CAPTCHAs are widely used due to their simplicity, where humans have to type displayed characters. The reCAPTCHA [2] is the most popular text CAPTCHA. Recent research in [9] addresses the importance of correctly configuring 2D text CAPTCHA parameters to maximize usability and security. They performed an extensive evaluation and user study, and provided new insights and details on the factors to improve usability and security of 2D text CAPTCHA. However, their focus was to mainly evaluate existing 2D text CAPTCHA systems. Much research including [8,20] has shown that it is very easy to break 2D text CAPTCHAs with text recognition and segmentation attacks. Therefore, 2D text CAPTCHAs have a fundamental limitation that can be defeated by the improved OCR and segmentation attacks. Hollow CAPTCHAs were proposed as alternatives, which use contour lines to form connected hollow characters and further aim to make segmentation and recognition difficult in a 2D space. However, [14] successfully broke a whole family of hollow CAPTCHAs.

Another popular CAPTCHA is an image based CAPTCHA that requires a user to classify and choose specific images. The Asirra CAPTCHA [13] is an image classification-task-CAPTCHA that for example, distinguishes a cat image from a dog image. However, the automated machine classification algorithm proposed by [15] can easily break this approach. Also, Sketcha [23] asks a user to rotate images to specific directions or positions. This is similar to our approach as users have to perform a rotation task. However, rotation in Sketcha is performed in a 2D space using rather simple image sets, which can be potentially defeated by a 2D image classification algorithm. The new reCAPTCHA (No CAPTCHA reCAPTCHA) includes an object classification task as well, where it asks a user to select all of the "cake", "cat", or "sushi", etc. images. However, [25] shows that new image based reCATPCHA can be broken with more than 70% accuracy using off-the-shelf deep learning package. In addition, animated CAPTCHAs, also known as moving-image (video) object recognition CAPTCHAs, have been proposed. However, [28] demonstrated a way to break animated CAPTCHAs with computer vision and motion tracking techniques using classifiers. Further, they proposed the new CAPTCHA based on an emerging image concept in [19]. However, it is still in the 2D domain. In fact, CAPTCHA characters can be displayed in a 3D space. However, recent work [21] demonstrated the step-by-step approach to break 3D characters with offline processing techniques. They claimed to decode the 3D text CAPTCHA with 92% accuracy.

3 3D CAPTCHA Design

We explore two types of 3D CAPTCHAs: *3D Texture CAPTCHA* and *3D Projection CAPTCHA*. In *3D Texture CAPTCHA*, we embed CAPTCHA characters directly to the surface of a 3D model as a part of textures using *UV* [5] mapping, where *UV* maps 2D textures to a 3D model. This allows adding various textures, backgrounds, and noise effects from the 2D text CAPTCHAs onto a 3D object. Users have to identify the surfaces that CAPTCHA characters are located. In the second 3D Projection approach, we project a 2D text CAPTCHA to a 3D model. In particular, we projected the 2D text CAPTCHA image from a specific view point towards a 3D object. Therefore, users have to identify a correct projection view point, in order to recognize CAPTCHA characters. Users can rotate a randomly placed 3D object in a 3D space to locate CAPTCHAs in a direct view. Once, users are able to locate the CAPTCHA, they can read and enter the CAPTCHA characters similar to 2D text-based CAPTCHAs.

3.1 2D Character Generation

Similar to current 2D text CAPTCHAs, we randomly generated 2D characters as an image file. The characters are read from left-to-right direction. We used the similar character sets recommended in [9] for lowercase letters, uppercase letters, digits, and non-confusable letters. In addition, font color, dot size, number of dots, number of lines, and line width can be used as configurable parameters. However, we did not consider these additional parameters. Instead, we capture those lines and dots as a part of texture in a 3D model. Since the font and color of CAPTCHA characters do not impact the usability and security according to [9], we only used the one font and one color for convenience in this work.

3.2 3D Object Generation

First, we created simple 3D models first using Unity [3] such as cube, circular cone, cylinder, triangular pyramid, hexagonal pyramid, etc. as base objects. Then, we combined and concatenated multiple 3D base objects to construct more complex synthetic shapes and polygons. Especially, we generated synthetic shapes to have irregular convex or concave surfaces to ensure that CAPTCHA characters are not easily detectable and recognizable. Also, this allows that a character can span over multiple surfaces so that it requires to rotate multiple times to completely recognize a character.

Further, these synthetic shapes are advantageous since these are new models that do not exist in the real world. Therefore, it is very difficult to train machine learning algorithms to perform meaningful prediction and classification because of a lack of a training set on the newly created 3D models. In addition, Unity provides more than 1,000 free 3D models to use. We collected freely available 3D object models that Unity provides such as a car, a piano, a chair, etc. Therefore, our 3D models include models that exist in real life as well as syntactically created objects. Some samples of syntactically generated 3D models and 3D models from real life are as the Figs. 1 and 4 respectively.

3.3 Approach1: 3D Texture CAPTCHA

We placed the generated 2D text CAPTCHAs to different 2D textures, where different types of fractal lines or background noise effects can be added to the texture. This has much more stronger impact than adding dots or lines in current 2D text CAPTCHAs. For texture mapping, we have following two different base texture sizes, (1) 1024×1024 and (2) 2048×2048 resolution (in pixels), in order to capture the granularity of a texture in a 3D model. Based on our empirical experiments on configuring different texture and model sizes, above two different texture resolution sizes suffice most of our needs. Once we generated a combined texture and CAPTCHA characters in 2D, we applied a UV mapping [5] to map a 2D texture in (u,v) coordinate onto a 3D object in (x,y,z) coordinate. The UV mapping assigns pixels in 2D image to surface on the polygon, by copying a triangle shaped piece of the 2D image map onto a triangle on a 3D object.

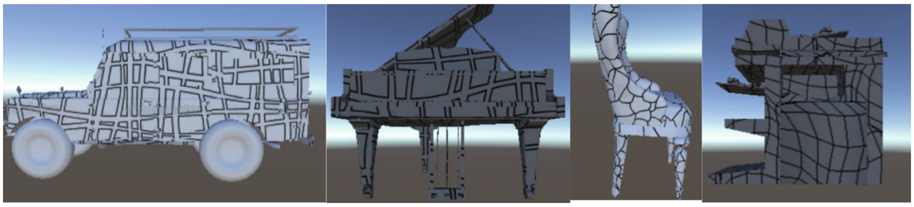


Fig. 1. Examples of 3D Texture CAPTCHAs

The texturing and *UV* mapping is a one-way transformation. Hence, after this step, it is nearly impossible to simply separate CAPTCHA characters from a 3D model. Figure 1 shows examples of 3D models with *UV* mapped textured from the previous step. Therefore, finally we created a 3D object with CAPTCHA characters embedded in its texture.

3.4 Approach 2: 3D Projection CAPTCHA

In this approach, we project CAPTCHA characters onto a 3D object as shown in Fig. 3. The main idea is to project a 2D text CAPTCHA from a specific angle. Hence, viewing from a slightly different view point (angle) would blur or occlude projected CAPTCHA characters. In Fig. 3, the projection point is shown as a red light source, which projects the 2D text CAPTCHA image toward a 3D object. Before projecting a 2D text CAPTCHA, we preprocessed the generated 2D text CAPTCHA image to be 64×64 pixels so that the 2D CAPTCHA can be smaller than the 3D object size. Further, we varied the alpha channel value [1] of 2D text CAPTCHA, where alpha channel value or compositing controls the transparency of the image from background. We set CAPTCHA characters to be transparent, while setting opaque for other background in the image as shown in Fig. 2. In Fig. 2, the left figure is the original 2D text CAPTCHA image. The right figure is the generated image from changing the alpha channel value so that background is black (opaque) while all characters are white (transparent). This alpha compositing functions a filter so that a light only passes through the character parts while the rest of opaque parts are completely blocked. Certainly, other background effects such as dots and lines can be added before this step similar to 3D Texture CAPTCHA to improve usability and security. Next, we placed a 3D object in $(x,\ y,\ z)$ coordinate as shown at the center in Fig. 3. In addition, various lighting effect such as directional lights, point lights, and spot lights with different color can be added from any 3D space coordinate, and further projected into the 3D object to make CAPTCHA recognition difficult. Furthermore, we can add and configure different shading effects at this step. Some examples of finally generated 3D Projection CAPTCHAs are shown in Fig. 4.

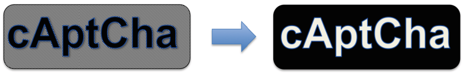


Fig. 2. Alpha channel value changed before Projection

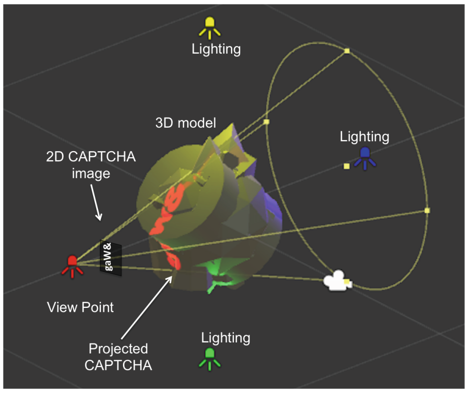


Fig. 3. 2D text CAPTCHA Projection onto a 3D model with different light sources (Color figure online)

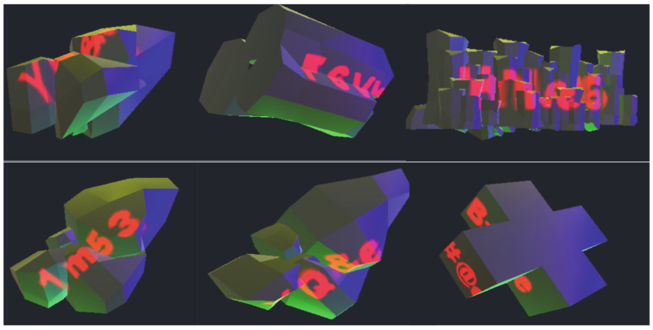
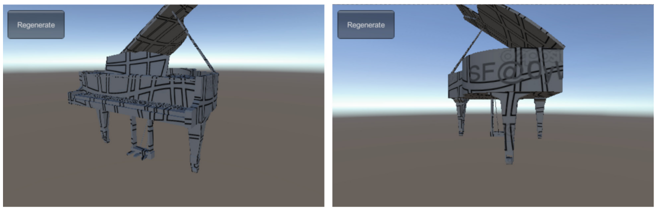


Fig. 4. Examples of 3D Projection CAPTCHAs

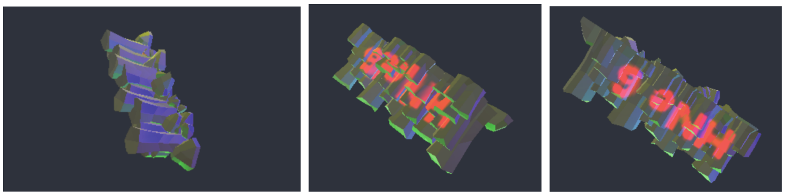
3.5 Final 3D CAPTCHA Generation

Finally, we generated 3D Texture and Projection CAPTCHAs by compiling objects, models, and textures we defined in the previous steps with the Unity 3D Web Player. The Unity Web Player [3] generates a binary to be completely played in the web browsers after installing the plugin. We developed the web-based user interface to move and rotate a 3D object in 360° in (x, y, z) direction with a mouse click. Following Figs. 5(a) and 6(a) show the generated 3D Texture and Projection CAPTCHAs, respectively. Also, we present screen captures of object rotations to identify the CAPTCHAs in each approach in Figs. 5 and 6. In the



(a) Initial placement (b) Locating the CAPTCHA

Fig. 5. 3D Texture CAPTCHAs to locate "SF@OyW"



(a) Initial placement (b) Rotate to find a CAPTCHA (c) CAPTCHA characters are in the direct view

Fig. 6. 3D Projection CAPTCHAs to locate "HNe6"

first 3D Texture approach, users have to rotate the piano object multiple times to locate CAPTCHA characters "SF@OyW" as shown in in Fig. 5(b).

In the second projection approach, users have to rotate a 3D object to view from a projection point as well as they need to adjust the precise viewing angle to recognize all CAPTCHA characters, "HNe6", as shown in Fig. 6(a)–(c). Furthermore, in the second approach, if users view from a slightly different angle such as in Fig. 6(b), the CAPTCHA characters appear blurred. Once users locate the CAPTCHAs, we provide a text box to enter CAPTCHA characters similar to reCAPTCHA.

4 Potential Attacks to 3D CAPTCHAs

We analyzed and conducted the following five realistic attacks against 3D CAPTCHAs: (1) Off-The-Shelf OCR attack, (2) Computer Vision Attack, (3) Client Side Code Attack, (4) Brute-Force based Incremental Object Rotation Attack, and (5) CAPTCHA relay attack for human solvers. The first attack is using OCR. Although OCR is getting better, current OCR technology focuses mainly on 2D object and character recognition. We tried the commercial OCR [6] to our 3D models and the 2D OCR performed extremely poorly against our systems. The 2D OCR by itself is useless to attack our systems. Therefore, we do not present the result with the OCR attack alone. Other more serious attacks are discussed in the following sections:

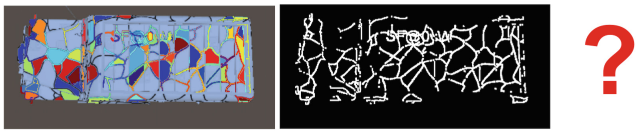
4.1 Computer Vision Attack

A bot has to recognize a 3D object first and rotate the object. Then, a bot should perform a series of segmentation and CAPTCHA recognition attacks similar to attack 2D text CATPCHAs. Currently, we are not aware of any approaches that can intelligently rotate objects, and detect characters in an automated way. Hence, in order to provide meaningful quantitative comparisons with computer vision attack, we relaxed our assumption significantly. We rotated an object and presented parts or all of CAPTCHA characters to be in the direct or near direct view so that the CAPTCHA recognition algorithm can directly attack. We defined the viewing angle, θ, to be the angle between users looking at the screen and the surface where the CAPTCHA characters were located in a 3D model. Therefore, $\theta = 0°$ was the direct view towards the CAPTCHA characters, effectively making 3D CAPTCHAs to the 2D text CAPTCHAs. We varied θ from $-45°$ to $45°$ so that all or part of CAPTCHA characters were shown. If θ is greater than $\pm 45°$, then CAPTCHA characters would not be seen to users.

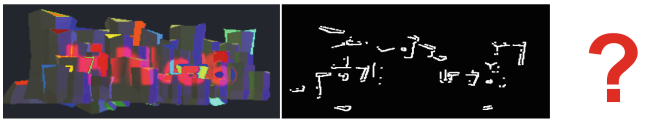
Then, we performed an offline computer vision attack under this relaxed assumptions. In particular, we used maximally stable extremal regions (MSER) region detector [11] to effectively detect regions, objects, and characters with modifications. The sequence of the proposed offline attack was as follows: (1) detect regions containing CATPCHAs, (2) segment out characters from a 3D model, and (3) perform optical character recognition (OCR). In particular, we used the Canny Edge Detector [10] to further segment the CAPTCHA characters after detecting the MSER region. Also, we filtered character candidates using connected component analysis. For the 3D Texture CAPTCHA, we used 6 different models and 2 different textures, and randomly generated characters. Also, in order to improve CAPTCHA detections, we zoomed-in the area where CAPTCHAs were located for offline processing. This scenario is the optimistic scenario, where a bot already figured out the 3D object rotation and CAPTCHA area identification, and further it could focus on the region that to improve CAPTCHA segmentation after removing noise.

Figures 7(a) and (b) are the examples of CAPTCHA characters presented in the direct view for 3D Texture and Projection CAPTCHA, respectively. The identified MSER region is shown in the first figure of Fig. 7(a). The second figure in Fig. 7(a) is the final CAPTCHA region detected by the proposed attack, and it cannot recognize the text. Next, we carried out the same attack to 3D Projection CAPTCHAs with 17 different models, where algorithms attempted to identify the CAPTCHA character region with MSER detector and the Canny detector in Fig. 7(b). However, in Fig. 7(b), it cannot not successfully identify the correct region and further is not able to carry out CAPTCHA recognition, even if the CAPTCHA is presented in the near direct view. On the other hand, we applied the same attack to the new text-based reCATPCHA. As shown in Fig. 7(c), the off-the-shelf computer vision algorithm can successfully detecting the CAPTCHA "12028".

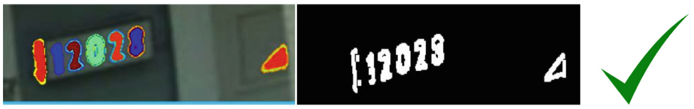
In Fig. 8, we provide the average CAPTCHA character recognition success rate from the proposed attacks. We consider an attack is successful only if all the



(a) Attack on 3D Texture CAPTCHA: MSER Region Detection and Recognition Failure



(b) Attack on 3D Projection CAPTCHA: MSER Region Detection and Recognition Failure



(c) Attack on New Text reCAPTCHA: MSER Region Detection and Recognition Failure

Fig. 7. Computer vision attack on each approach

given CAPTCHA characters are correctly recognized from the viewing angle, θ. For example, it is possible that only 2 characters are visible if $\theta = -30°$, while 5 characters can be shown if $\theta = 0°$. In order to compare, we manually rotated objects to be at $\pm45°, \pm30°$, $\pm15°$ and $0°$ from the direct view point, and captured those images, and inputted those images into our proposed attacks. The detection results are shown in Fig. 8. From 50 manually rotated and captured images, overall on average 48.9% and 31.5% of all of given CAPTCHAs were successfully detected for 3D Texture and Projection CAPTCHAs, respectively. At $\theta = 0°$, we zoomed and focused the CAPTCHA region area to improve detection. Then, the detection success rate increased to 60% and 64.7% for 3D Texture and Projection CATPCHAs, respectively. This demonstrates that even viewing from the precise and best angle, a computer vision algorithm does not achieve the high success rate as found in the 2D text CAPTCHAs. Also, it was interesting to observe that, if angle was improved from $\pm30°$ to $\pm15°$, the detection rate was not improved at all. This is due to the fact that our CATPCHAs span over multiple surfaces and require to have multiple views to locate all CAPTCHA characters. Hence, our design is resistant to current 2D text CAPTCHA attack in several different ways. If $\theta \neq 0°$, 3D Projection CAPTCHA was more difficult to detect than 3D Texture approach. However, if $\theta = 0°$, then 3D Texture CAPTCHA was slightly more difficult to detect than 3D Projection CAPTCHA.

Also, we obtained 50 different new text-based reCAPTCHA and performed the same attack. We found out that 82.9% of new text reCAPTCHA were completely detectable with the simple automated computer vision attacks. The reCATPCHAs detection success rate was much higher than ours in the most

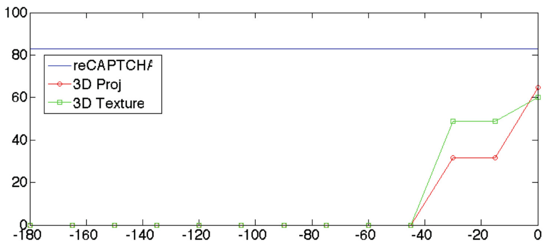


Fig. 8. Viewing angle (θ) vs. Detection success rate with automated computer vision attacks(%)

optimum scenario: only if a machine can perfectly rotate, identify, and focus on the CAPTCHA area in a 3D model (finding $\theta = 0°$) as humans do, then it can achieve the highest success rate. Therefore, for the same attack, our approach is much stronger than the new text reCAPTCHA.

4.2 Client Side Code Attack and Binary Analysis

Although the WebGL [17] provides the similar 3D graphic capabilities, we specifically did not use the WebGL due to the potential client side code attack. Since WebGL relies on the client side JavaScript to render 3D object, textures, and further CAPTCHAs, it is possible to parse the WebGL that describes 3D objects, 2D textures, and extract the 2D text CAPTCHAs. On the other hand, Unity generates a Web Player binary that includes all 3D objects, textures, and CAPTCHAs from the backend and only plays through the web browser. Therefore, it is difficult to reverse engineer and parse the CAPTCHAs from the generated binary. Although it is not impossible to decompile and obfuscate the Unity script using a tool such as [4], we generate a new CAPTCHA and texture information from the backend and the client-side web player calls backend code to retrieve the detailed CAPTCHA and 3D information. Hence, even though attackers are able to decompile the binary, the CAPTCHA, 3D models, and textures information are not stored in the client side. Our approach is much more robust against the client side code attack compared to other 3D development platforms that heavily use client side code such as the WebGL.

4.3 Brute-Force Based Incremental Object Rotation Attack

It is possible that the automated system can incrementally rotate a 3D object until to detect the characters regions with high probability and try to identify characters with OCR types of applications. This Brute-force approach might take a lot of iterations but does not require any intelligence in understanding unseen synthetic 3D objects. If attackers have enough computing power, this Brute-Force based Incremental Object Rotation Attack can be a serious to our system.

Table 1. Human behavior modeling on 3D CATPCHAs

Modeling parameters	Avg	Median	std
Num. of clicks	3.31	2	0.0878
Click distance in pixels	150.26	126.5	0.0035

However, first, our textures and uneven surface make accurate detection difficult as we use different background noise and lighting effect as we observed from computer vision attack. Second, we measured the average time, and the number of clicks needed to solve CAPTCHAs from 90 participants as shown in Table 1. In order for this attack to be successful, a bot has to completely mimic human behavior and we observed that humans are not incrementally rotating objects. Therefore, our defense is to compare the average solving time and human click behavior with this attack, where each rotation requires a mouse click. Therefore, (1) if it takes far longer or shorter to solve or (2) significantly exceeds the number of clicks usually needed for humans, we can classify this as an attack and reject. Using gathered statistic on typical human behaviors on solving 3D CAPTCHA as a prior information, we believe we can easily mitigate and reject the brute-force based incremental object rotation attacks.

5 Conclusion and Future Work

We present two different 3D CAPTCHA mechanisms using 3D objects, projection, and texture techniques. Although our concept is simple, we provide much stronger protection against various existing attacks, which can easily break current 2D text CAPTCHAs. The future work is to conduct user study to evaluate the usability and security trade-offs among different CAPTCHA mechanisms.

Acknowledgements. We would like to thank Ulrich Neumann, Michael Zyda, and Jelena Mirkovic for providing helpful comments and feedback.

References

1. Alpha compositing. http://en.wikipedia.org/wiki/Alpha_compositing
2. reCAPTCHA. https://www.google.com/recaptcha/intro/index.html
3. Unity. http://unity3d.com/
4. Unity 3D obfuscator. http://en.unity3d.netobf.com/unity3d_decompiler
5. UV mapping. http://en.wikipedia.org/wiki/UV_mapping
6. AABBYY OCR software. http://www.abbyy.com/
7. Aldoma, A., Tombari, F., Stefano, L., Vincze, M.: A global hypotheses verification method for 3D object recognition. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7574, pp. 511–524. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33712-3_37

8. Bursztein, E., Martin, M., Mitchell, J.: Text-based CAPTCHA strengths and weaknesses. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, pp. 125–138. ACM (2011)
9. Bursztein, E., Moscicki, A., Fabry, C., Bethard, S., Mitchell, J.C., Jurafsky, D.: Easy does it: more usable CAPTCHAs. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, pp. 2637–2646. ACM (2014)
10. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 679–698 (1986)
11. Chen, H., Tsai, S.S., Schroth, G., Chen, D.M., Grzeszczuk, R., Girod, B.: Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In: 2011 18th IEEE International Conference on Image Processing (ICIP), pp. 2609–2612. IEEE (2011)
12. Cui, J.-S., Mei, J.-T., Zhang, W.-Z., Wang, X., Zhang, D.: A CAPTCHA implementation based on moving objects recognition problem. In: 2010 International Conference on e-Business and e-Government (ICEE), pp. 1277–1280. IEEE (2010)
13. Elson, J., Douceur, J.R., Howell, J., Saul, J.: Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In: ACM Conference on Computer and Communications Security, pp. 366–374 (2007)
14. Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., Yan, J.: The robustness of hollow CAPTCHAs. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 1075–1086. ACM (2013)
15. Golle, P.: Machine learning attacks against the asirra CAPTCHA. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 535–542. ACM (2008)
16. Lowe, D.G.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision 1999, vol. 2, pp. 1150–1157. IEEE (1999)
17. Marrin, C.: WebGL specification. Khronos WebGL Working Group (2011)
18. Meutzner, H., Nguyen, V.-H., Holz, T., Kolossa, D.: Using automatic speech recognition for attacking acoustic CAPTCHAs: the trade-off between usability and security. In: Proceedings of the 30th Annual Computer Security Applications Conference, pp. 276–285. ACM (2014)
19. Mitra, N.J., Chu, H.-K., Lee, T.-Y., Wolf, L., Yeshurun, H., Cohen-Or, D.: Emerging images. ACM Trans. Graph. (TOG) **28**, 163 (2009). ACM
20. Mori, G., Malik., J.: Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, Proceedings, vol. 1, pp. 1–134. IEEE (2003)
21. Nguyen, V.D., Chow, Y.-W., Susilo, W.: Breaking a 3D-based CAPTCHA scheme. In: Kim, H. (ed.) ICISC 2011. LNCS, vol. 7259, pp. 391–405. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31912-9_26
22. NuCaptcha - most secure and usable CAPTCHA. http://www.nucaptcha.com/
23. Ross, S.A., Halderman, J.A., Finkelstein, A.: Sketcha: a CAPTCHA based on line drawings of 3D models. In: Proceedings of the 19th International Conference on World Wide Web, pp. 821–830. ACM (2010)
24. Shirali-Shahreza, M., Shirali-Shahreza, S.: Motion CAPTCHA. In: 2008 Conference on Human System Interactions, pp. 1042–1044. IEEE (2008)
25. Sivakorn, S., Polakis, I., Keromytis, A.D.: I am robot: (deep) learning to break semantic image CAPTCHAs. In: IEEE European Symposium on Security and Privacy (EuroS&P) (2016)

26. Vedaldi, A., Fulkerson, B.: VLFeat: an open and portable library of computer vision algorithms (2008). http://www.vlfeat.org/
27. Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003). doi:10.1007/3-540-39200-9_18
28. Xu, Y., Reynaga, G., Chiasson, S., Frahm, J.-M., Monrose, F., van Oorschot, P.C.: Security and usability challenges of moving-object CAPTCHAs: decoding codewords in motion. In: USENIX Security Symposium, pp. 49–64 (2012)