Received 19 December 2014
Revised 12 July 2015
Accepted 13 July 2015

Gesture-based animated CAPTCHA

Artem Shumilov and Andrey Philippovich
Research and Education Centre of Infocognitive Technologies, Moscow State Engineering University, Moscow, Russia

Abstract

Purpose – This paper aims to describe the development of a new experimental CAPTCHA (completely automated public turing test to tell computers and humans apart), which is supposed to provide better protection against spam bots compared to the existing captcha solutions.

Design/methodology/approach – In the new CAPTCHA, the authors are using animated images of hand gestures to represent letters and numbers. A lot of people use sign language as their primary means of communication, but an automatic algorithm able to reliably recognize such gestures from videos without any additional data has yet to be developed.

Findings – The experiment showed that at first, it takes time for people who do not use sign languages in their everyday lives to adapt to the new CAPTCHA, but after several successful recognitions, they have no more trouble with it than with a typical captcha implementation.

Originality/value – The paper shows that people with little to no knowledge of sign languages can still recognize gestures on video relatively fast. Therefore, a gesture-based implementation can be used not only on websites aimed at sign language speakers but also as a general-purpose captcha service.

Keywords Innovation, Computer security, Access control, Client-server computing

Paper type Research paper

Introduction

In spite of spam's low conversion rate (Kanich *et al.*, 2008), it is still a popular way of making money (Gudkova, 2014). To prevent the unsolicited remarks left by spambots from appearing on their websites, webmasters often use captchas (completely automated public turing test to tell computers and humans apart; von Ahn, 2011). Unfortunately, it is possible to solve them automatically. For example, ReCaptcha, one of the most popular modern captchas, has been broken by a neural network (Hof, 2013). It was neither the first nor the last captcha to be broken by such a system, for that information processing paradigm is especially good at solving recognition-related problems (Vylomova *et al.*, 2014). However, artificial intelligence (AI) systems have not achieved similar results in the field of gesture recognition yet (Rafiqul and Noor, 2012).

So, is it possible to create a good gesture-based CAPTCHA? The perfect captcha should not only be hard to pass for robots but also be easy for humans and should possess several characteristics described in paper (Bushell, 2011). Research at the premises of scientific-educational cluster CLAIM Philippovich (2013), Shumilov and Philippovich (2014), An Animated CAPTCHA that uses Hand Gestures, 2014 Shumilov and Philippovich (2016), Using 3D Animated Hand Gestures to Create a New Type of CAPTCHA) revealed that gesture recognition is easy for people familiar with manual communication techniques. In fact, sign languages have all the features of a natural

Information & Computer Security
Vol. 24 No. 3, 2016
pp. 242-254

2056-4961
DOI 10.1108/ICS-12-2014-0082

language (Sandler and Lillo-Martin, 2001). Therefore, it has been decided to develop an experimental gesture-based animated CAPTCHA.

This paper strives to describe the process of creating a captcha that requires the user to recognize hand gestures to pass the test. We start with a brief introduction of the basic algorithm most of modern captchas use to determine whether the test subject is a human or not, discuss the user roles such a system usually has and suggest one more role necessary for state-of-the-art systems. Then, we move on to the specific problems faced by gesture-based and, to a lesser extent, animated captchas in general. We describe our gesture renderer, discuss different video rendering approaches and compare the most popular online video formats from the perspective of creating short videos and sending them over the internet. In the end, we report the results of the experiment that was supposed to reveal how long it usually takes people to recognize a simplified gesture-based CAPTCHA when they see it for the first time and how fast they get used to it.

Toward the gesture-based CAPTCHA service

Some of challenge/response systems not only provide protection against spam but also perform other useful operations, for example, help recognize scanned texts and house numbers. However, their main task is to tell a website whether the user is a human. There are a lot of different captcha services, and most of them work according to the algorithm shown in Figure 1.

There are three different machines performing operations. The client device belongs to a visitor of the website. It can be anything that is able to run a more or less modern browser and has it installed. Webmaster provides an application server, the computer that handles all the procedures necessary for the website. The last device generates the tests and checks the responses. It usually belongs to the company that provides the captcha service.

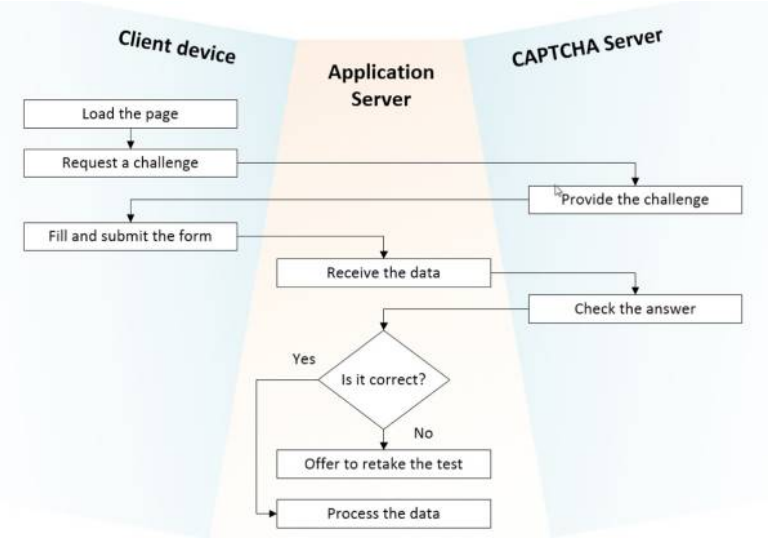

Figure 1.
The way captchas work

The application server and the CAPTCHA server are usually two different devices because creation of test images requires a considerable amount of computing resources and, sometimes, specialized hardware. It highly depends on the type of installed captcha and popularity of the website, so it can be almost imperceptible and nearly paralyze the work of the server and, therefore, website. Usually, website owners prefer to avoid dealing with the potential performance problem and use third-party services to get the job done.

User classification

Three types of users, each one with its own specific functions, can be distinguished from the standard captcha functioning algorithm (Figure 1):

(1) The *administrator role* represents people who control and maintain the captcha server, as well as keep an eye on the workload statistics, to be able to upgrade the server in time.

(2) The *surfer class corresponds* to a visitor of a website protected by the system. Surfers almost do not interact with the service directly, and most of them do not even know that the page they are looking at and the challenge/response test on that page are provided by different companies. However, when they follow a link that leads to a protected webpage, their browsers request a vidget from the captcha server to display it.

(3) The *webmaster* is the last standard class. These people should be able to register and set up an account, indicate the websites that have to be protected and choose specific settings for each one of them if needed. For example, it may be possible to select a set of gestural alphabets, adjust the size of the plug-in, select a hand model that will sign the gestures and pick a background and some rendering parameters. The structure of the accounting is shown in Figure 2. As the abilities to pick the size and the alphabet require to generate the videos for each website individually, a webmaster can be allowed to select a lot of parameters and even create new gestures that are to be shown nowhere but the specified websites.

An additional role, the *analyst*, can be distinguished for the gesture-based CAPTCHA service. As the proposed type of CAPTCHA is new, it is very important to do research on recognition difficulty, success rate and speed of different gestures especially by the users who are not familiar with sign languages. It will allow to improve the quality of the service and to create new gesture alphabets. During a later stage of implementation, the functions of analysis and adjustment can be standardized and partially put under care of the webmaster.

Gesture editor

One of the sailient features of the gesture-based CAPTCHA is considerable customization capability. Aside from eye-candy adjustments, it is possible to create individual gestures for each website, which will force spammers to teach their algorithms to recognize the test videos from each particular website that may sometimes be not worth the effort.

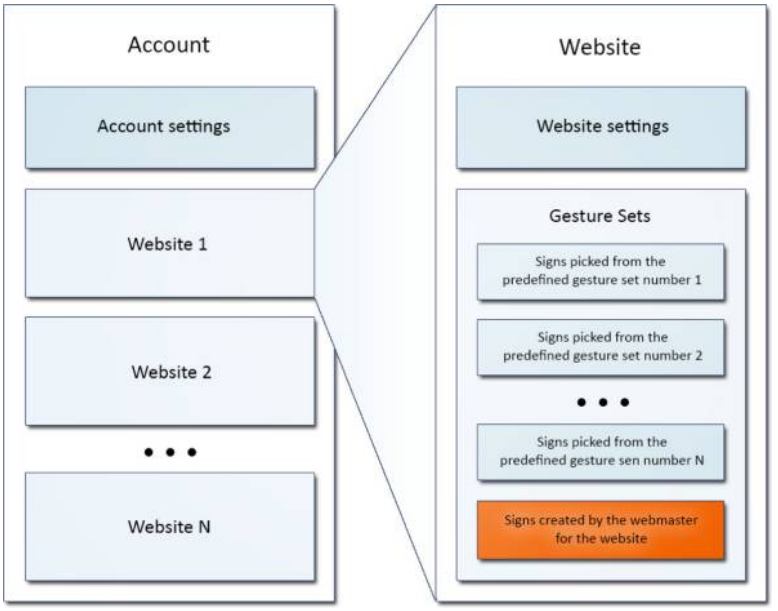

Figure 2.
Webmaster account structure

Creation of a realistic hand model

That kind of an editor requires the ability to adjust the pose of the hand model and see the changes in real time. Besides, if possible, it should not force webmasters to install any additional software. To solve the problem, the following technologies were selected: HTML5, that allows three-dimensional (3D) graphics to be rendered using hardware acceleration inside its canvas element without the need to install any additional plug-ins; WebGL, a JavaScript rendering application programming interface (API); and three.js, a cross-browser high-level JavaScript API/library that significantly facilitates the use of WebGL.

The developed application uses a hand model represented by triangular polygons. Every vertex of the model is assigned to one or more bones, so it is possible to programmatically adjust the pose of the hand on the fly by rotating the bones around their heads.

Most of the modern 3D editors, including Blender, the one that was used to make the hand, allow not only creating and rigging but also animating a model. It is planned for the future to write a plug-in that will make it possible to create gestures by means of the most popular 3D editors such as 3D Max, Maya, etc. and export them to be used in the application gesture format so that webmasters who are familiar with those editors could create gestures in the environment they like and do not need to get used to. The movement of a signing hand can be divided into two parts:

(1) Movements in the process of showing a gesture; and

(2) Transitions between gestures.

If a model is required to show only one sign, it is not that hard to create both parts in an editor. But, if it is necessary to demonstrate a sequence of gestures, every one of them

should have transitions from and into all the rest. The more gestures already exist in the database, the more time it would take to add a new one. Therefore, the transitions should be calculated automatically.

Creation of a hand model capable of realistically signing whatever gesture is wanted seems to be rather easy, but, in spite of that, it includes several difficulties. Such parameters as the length of the bones, the position and range of freedom of the joints had to be revealed. Estimation of the most of them is not a trivial task. However, a lot of research in the field of anthropometrics has already been done, so it was possible to use some results. The paper (Buryanov and Kotiuk, 2010) reports about an analysis of the length of the bones, the results of which were used during the process of making the hand model. The armature is shown in Figure 3. Estimations of the ranges of motion of the joints have been conducted and described as well Bain *et al.* (2014), Lowe (2006), Joaqui′n L. Sancho-Bru (2011). The results of them were used too, but, sometimes, it was necessary to slightly adjust them to avoid intersections between different parts of the model.

The hardest thing was to estimate the position of carpometacarpal joints (M1-M5 in Figure 3) and metacarpophalangeal (MCP) joints (P1-P5 in Figure 3). Given the length of the metacarpal bones, the latter resulted in the position of the related MCP joint and the direction of the metacarpal (depicted in dark blue in Figure 3) bone. It was necessary to create about 30 gestures adjusting those variables in the process to collect statistics and find the values that allowed showing all the gestures without repositioning.

Structure of the gesture information

There are two different types of signs: postures and gestures. Postures are static and do not imply any repositioning of the model in the process of showing one of them. That is to say, signing them does not involve any movement but transitions. Gestures, on the other hand, require the model to move in cycles while showing one single sign, for example, waving the hand or curving a finger. Otherwise, the meaning can be completely different. Therefore, in the application, gestures are represented by a sequence of postures, each of which has to be shown at a specified time. This approach is also known as time-based animation.

A posture is represented by a position and rotation of the model as a whole and a set of relative angles between bones. A designer can only move the model and rotate bones; the position of a bone can only change as a result of rotating one of the parent bones. Also, each bone has some limitations in the range of motion that allow it to rotate only

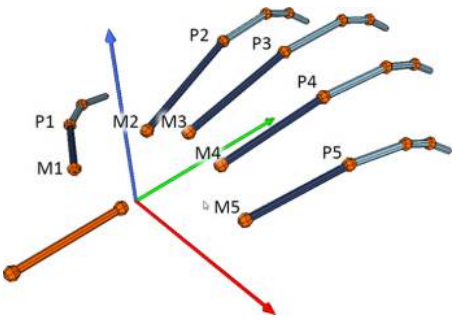

Figure 3.
Hand armature

around some axes that are known for each bone as a result of the statistical data analysis mentioned in the previous section. Therefore, the data necessary to describe a posture consist only of timings, angles, indexes of the first and the last frames in the loop and the position of the hand. Figure 4 illustrates an Extensible Markup Language (XML) file that fully describes a gesture that consists of five postures, where the frames from two to four are cycled in the loop. In fact, the application uses JavaScript Object Notation (JSON) format over XML to transfer the data, but XML allows to show the information more graphically.

Different video rendering approaches

Transferring video over the internet requires a rather high connection speed, and its rendering consumes a lot of computing power. Therefore, it is necessary to compare different approaches to rendering, inspect their positive and negative sides and choose the one that fits the purpose best.

```
<!-- Indexes of the first and last frame in the loop
    and timing of each frame in milliseconds -->
<time loopStart='2' loopEnd=='4'>0,700,1200,2300,3000</time>
<!-- Hand position and rotation -->
<hand xPos='0,11,22,33,44' yPos='0,11,22,33,44' zPos='0,11,22,33,44'
    xRot='0,11,22,33,44' yRot='0,11,22,33,44' zRot='0,11,22,33,44'></hand>
<!-- Rotation of each finger bone -->
<finger>Index
    <bone pitch='0,11,22,33,44' yaw='0,11,22,33,44'>Proximal</bone>
    <bone pitch='0,11,22,33,44'>Intermediate</bone>
    <bone pitch='0,11,22,33,44'>Distal</bone>
</finger>
<finger>Middle
    <bone pitch='0,11,22,33,44' yaw='0,11,22,33,44'>Proximal</bone>
    <bone pitch='0,11,22,33,44'>Intermediate</bone>
    <bone pitch='0,11,22,33,44'>Distal</bone>
</finger>
<finger>Ring
    <bone pitch='0,11,22,33,44' yaw='0,11,22,33,44'>Proximal</bone>
    <bone pitch='0,11,22,33,44'>Intermediate</bone>
    <bone pitch='0,11,22,33,44'>Distal</bone>
</finger>
<finger>Pinky
    <!-- It is possible to pitch up and down pinky metacarpal bone -->
    <bone pitch='0,11,22,33,44'>Metacarpal</bone>
    <bone pitch='0,11,22,33,44' yaw='0,11,22,33,44'>Proximal</bone>
    <bone pitch='0,11,22,33,44'>Intermediate</bone>
    <bone pitch='0,11,22,33,44'>Distal</bone>
</finger>
<!-- Thumb has an unusual set of bones -->
<thumb>
    <bone pitch='0,11,22,33,44' yaw='0,11,22,33,44'>Metacarpal</bone>
    <bone pitch='0,11,22,33,44'>Proximal</bone>
    <bone pitch='0,11,22,33,44'>Distal</bone>
</thumb>
```

Figure 4.
Structure of the gesture information

Server side rendering

This is the first approach that comes to one's mind. It has a very important advantage: the server sends a rendered video to the client, so the only way to know what is shown on it is to recognize it, that is to say, to solve a problem, which is very hard for bots according to the assumption that was made in the Introduction. The model used to sign is stored in the database and does not have to be transferred anywhere. The algorithm that moves the bones and renders the scene is also stored on the server and is not accessible to hackers.

The most important disadvantage of the approach is high computing power requirements of video rendering. In spite of the fact that the activity of users depends on the time of day, and, therefore, the server can use off-hours to render additional videos proactively to lower the workload of peak hours, it significantly increases hardware requirements to the server (Figure 5).

Client side rendering

Another possible way to get the job done is client side rendering, for example, using the hardware of a client device to render the video. As unlike the server, it has to render only one relatively short video, the hardware requirements are not that high, so even smart phones would be able to do it. The server has to randomize the sequence of symbols that the user is going to be required to type, encrypt it and pass to the client. Aside from the obvious risk of decryption, it is worth mentioning that the technologies used to create the gesture editor can not be utilized here because JavaScript code is not protected at all and can be easily read by anyone (Figure 6).

Another disadvantage of that approach is the necessity to pass the model to the client device because, sometimes, its size is bigger than the one of a rendered video. Nowadays, there are still places on the globe where it is impossible to get fast internet connection

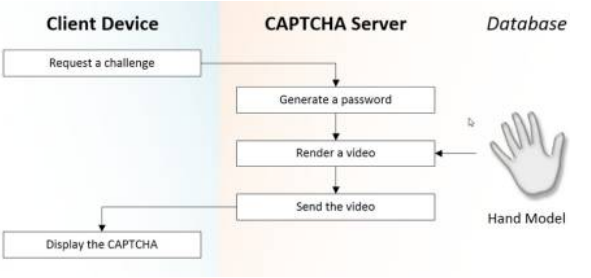

Figure 5.
Server side rendering

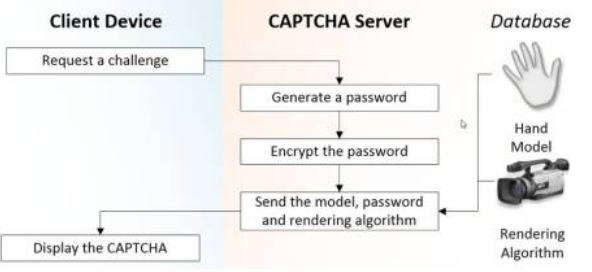

Figure 6.
Client side rendering with the cache disabled

even using a desktop; besides, a lot of people browse the Web on mobile devices and have to not only deal with slow connection speed but also pay for every megabyte of traffic. The problem can be partially solved if the user has not turned the Web cache off and, therefore, does not have to download the model every time a test needs to be passed. In that case, the only data that have to be transferred are the encoded sequence of symbols, which solves the problem of network transmission but does not help with the one of decryption (Figure 7).

Combined rendering

The previous two methods can be combined, letting the server generate parts of the video related to gestures and leaving rendering the background to the client device, as it shown in Figures 8 and 9.

This approach combines some advantages and disadvantages of the preceding ones: it sometimes will take less time to render parts of the video on the server, but it will also facilitate recognition of the video because it will be a lot easier to distinguish the outline of signing hand models. It is much less vulnerable to hacking than pure client side rendering and also sometimes faster than that rendering on the server. Though it can be even slower, it depends on the size of the resulting files.

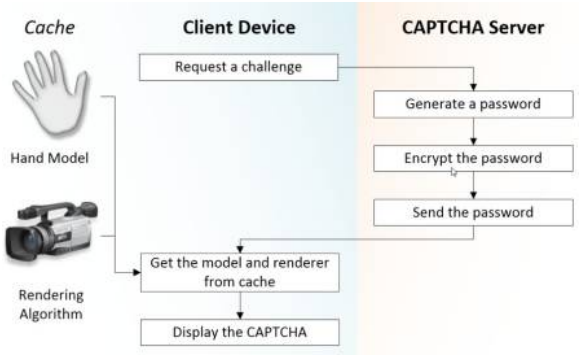

Figure 7.
Client side rendering with the cache enabled

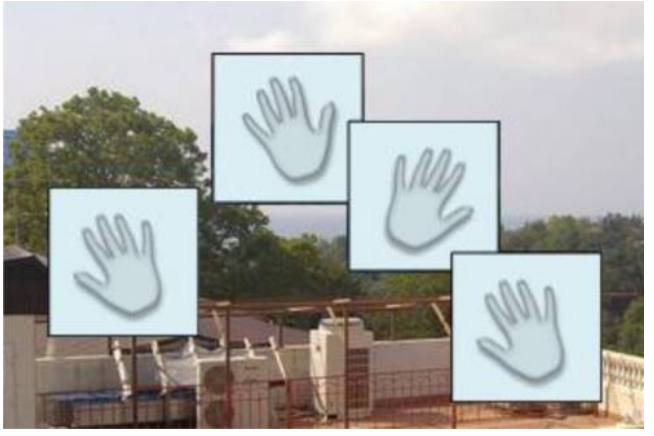

Figure 8.
Combined rendering result

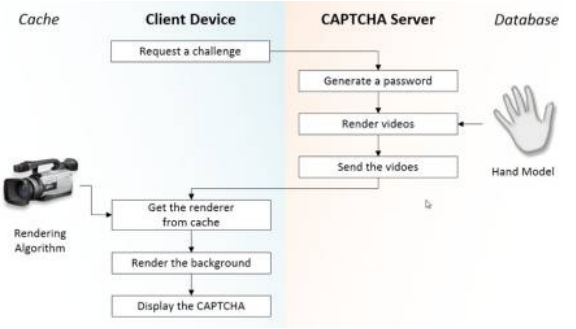

Figure 9.
Combined rendering

That approach seems to be the most tempting if the background is pre-defined and known to a hacker. For example, if an advertisement is used as a background, it would not be hard to distinguish the outlines even on a server side rendered video.

Video format comparison

Out of the three approaches described in the previous section, server side rendering seems to be the most reasonable, at least at the first stages of implementation. The combined rendering might be used later, especially in the situations when it actually gives some performance benefit. Using the client device to render the video is very unprotected and negates the main advantage of the gesture-based CAPTCHA, allowing computing of the symbol sequence without video recognition.

Now, it is necessary to pick a video codec that will be used to compress videos. The analysis of the dependence of the size of a video on the resolution and used codec was performed for the four most common video file formats (Grula, 2010) (AE Templates Marketplace, 2013). It was decided to exclude the audio video interleaved (AVI) format because it generates unacceptably big files even with the resolution of 64 × 64 pixels. The results are shown in Figure 10. The size of the hand model that has to be transferred over the internet in case of the client side rendering is also represented. It is easy to see that client side approach is not only unsafe but also requires more data to be transferred, unless the size of the video is enormously big for a captcha.

The time it took the server to render each of the videos was also measured and is represented in the form of a diagram (Figure 11). The time costs of making videos of the

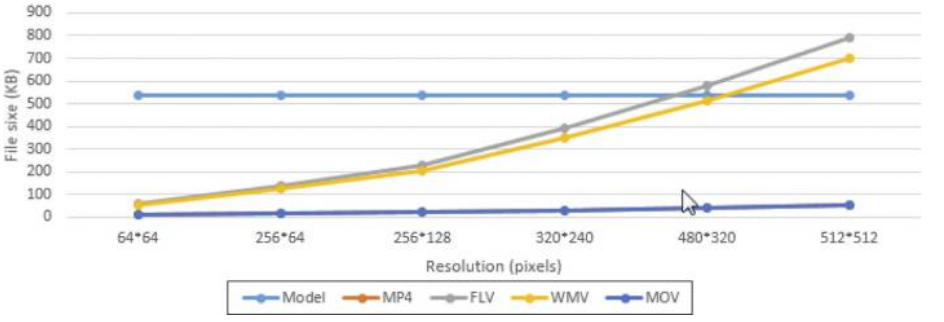

Figure 10.
Dependence of the file size on the format and resolution

same size with different codecs differ only slightly. Creating small videos do not provide any significant benefit over relatively big ones, and, considering that several of them should be created instead of a one of a bigger size, it actually requires more computing power per test.

Manual recognition rate of the gesture-based CAPTCHA

To estimate how hard it is for people to pass the gesture-based test, an experiment was performed. The CAPTCHA signing four numbers from 0 to 9 was shown to participants who were not familiar with gesture languages. Every one of them had to successfully recognize it 30 times. They were able to open a help page that clarified the meanings of the gestures, but the entire screen was blocked so they could not look at the CAPTCHA and the help at the same time. The goal of the test was to measure the average time it takes for a person to recognize the four gestures, the amount of mistakes made in the process and the time people spend on the help page and ascertain which gestures out of the ten are the hardest to recognize.

The screenshots of the task and the help page can be found in Figure 12. It is worth mentioning that the help page that shows the meanings of the gestures cannot help spambots recognize the video because the models on the video are not stationary. During the test, they were constantly transitioning between signing the fist and whatever gesture they were to show and rotating around the vertical axle.

The results of the tests are represented in Figures 13 and 14. At first, the testers had a lot of trouble solving the CAPTCHA, but, approximately after 17 successful recognitions, passing the test stopped taking them more than 12 s on average. This is a good result, as the most of the modern CAPTCHAs take from 8 to 12 s to pass as well. The gestures that represent the numbers 1, 2, 4 and 5 proved to be the easiest to recognize for the testers. It was a bit harder for them to distinguish the sign that stands

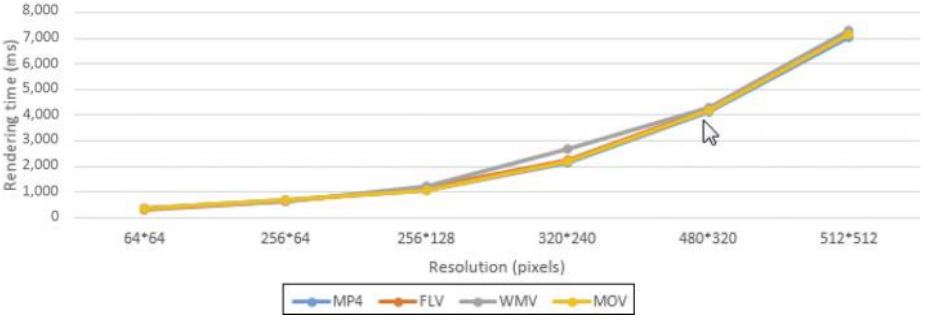

Figure 11. Dependence of the rendering time on the format and resolution

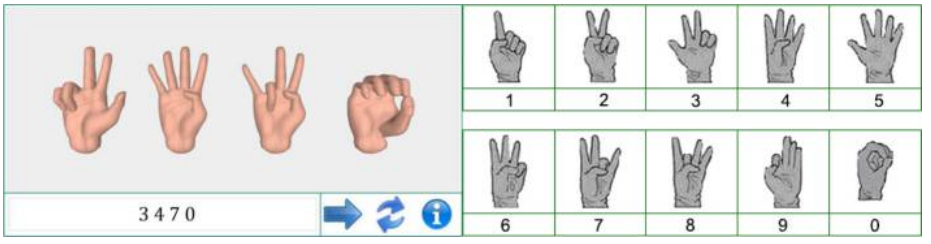

Figure 12. The CAPTCHA widget and help information that were used during the test

for 0 because it is slightly less common but still pretty obvious. The gesture that means "3" caused more trouble, probably because it looks similar to the one that stands for 6. The signs that represent the numbers from 6 to 9 appeared to be the hardest to recognize. The reason is that people do not use these gestures in everyday life and, therefore, do not know their meanings.

According to the results of the test, it takes an unprepared person approximately 2 min to successfully recognize the gestures, probably because people are not familiar with gesture languages. The fact that after several successful recognitions people stopped having troubles backs up that theory. The help page should have helped to overcome that obstacle; most likely, it could use some improvements. The most obvious idea is to duplicate the video on that page, so the person can have it opened all the time. It also might be helpful to allow the user to pause and roll back the animation. Another way to improve the help page is to represent the tips as 3D figures instead of images; this will allow the user to move the camera and look at the gestures from different points of view, but it would not be helpful for bots, because rendering takes a significant amount of computing power.

Conclusion

The conducted experiments have shown that the gesture-based CAPTCHA is not impossible to recognize even for people who are completely unfamiliar with sign languages, but it could use some improvements. As an algorithm able to automatically recognize hand gestures from a video without any depth data with a high rate of success

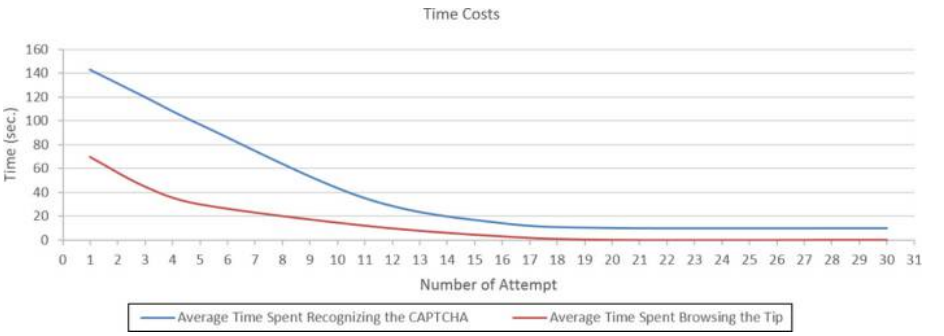

Figure 13.
The amount of time it takes for an unprepared person to solve the CAPTCHA

Figure 14.
The average number of mistakes made in the process

has not been created yet, using such signs to protect websites from spam can be a good idea.

Nowadays, a lot of people use mobile devices; sometimes, it is not convenient to use the most common input methods such as keyboards and touch screens on the move. Implementation of the gestural CAPTCHA can help do one more step for the creation of a new, gesture-based human–computer interaction system that would be able to facilitate the use of computers, especially by people who have trouble hearing and seeing things.

The number of people with such disabilities is relatively high, and it is important to provide them with means to communicate and work without having problems because of their impairments. Gesture-based interaction systems in general and the CAPTCHA in particular will teach people some signs; even though knowing some signs and being able to use a sign language are two very different things, it will ease communication with those of us who cannot use spoken languages by a great deal. Besides, sometimes, it is just more convenient to sign than to speak, for example, in a noisy environment.

The gesture-based CAPTCHA does not require any distortions to provide a good level of protection against spambots. The video can be as big as a webmaster wants it to be, and the approach itself is very different from the most common text-based captchas. All these reasons combined mean that it can be used as a backup system in case the user cannot solve another CAPTCHA. Currently, developers use audio CAPTCHA for the purpose that is not only inconvenient for people with disabilities but also requires the user to have speakers or headphones and sometimes makes unwanted noises.

In spite of the fact that the testing has shown that hardware requirements for the server are going to be relatively high and recognizing gestures is not easy for people who do not use sign languages on daily basis, the approach can prove itself useful. The disadvantages can be decreased by developing a more specific rendering algorithm that will solve that particular problem more effectively, using the database to store transition data instead of computing them on the fly every time they are required and providing a more direct and vivid help screen.

References

AE Templates Marketplace (2013), "What you need to know about five most common video file formats", available at: www.motionelements.com/blog/articles/what-you-need-to-know-about-the-5-most-common-video-file-formats (accessed 6 June 2014).

Bain, G., Polites, N., Higgs, B., Heptinstall, R. and McGrath, A. (2014), "The functional range of motion of the finger joints", *The Journal of Hand Surjery*, Vol 40 No. 4.

Buryanov, A. and Kotiuk, V. (2010), "Proportions of hand segments", *International Journal of Morphology*, Vol. 3 No. 28, pp. 755-758.

Bushell, D. (2011), "In search of the perfect CAPTCHA", available at: www.smashingmagazine.com/(2011)/03/04/in-search-of-the-perfect-captcha/ (accessed 8 November 2014).

Grula, L. (2010), "The basics of web video file formats and video containers", available at: www.reelseo.com/basics-web-video-file-formats-video-containers/ (accessed 28 May 2014).

Gudkova, D. (2014), "Kaspersky security bulletin: spam evolution (2013)", available at: http://securelist.com/analysis/kaspersky-security-bulletin/58274/kaspersky-security-bulletin-spam-evolution-2013/ (accessed 16 October 2014).

Hof, R. (2013), "AI startup vicarious claims milestone in quest to build a brain: cracking CAPTCHA", available at: www.forbes.com/sites/roberthof/(2013)/10/28/ai-startup-vicarious-claims-milestone-in-quest-to-build-a-brain-craking-captcha/ (accessed 2 October 2014).

Pérez-González, A., Mora, M.C., León, B.E., Vergara,M., Iserte, José L., Rodríguez-Cervantes, P.J., Morales, A., and Joaquín L. Sancho-Bru. (2011), "Towards a realistic and self-contained biomechanical model of the hand", available at: www.intechopen.com/books/theoretical-biomechanics/towards-a-realistic-and-self-contained-biomechanical-model-of-the-hand (accessed 1 November 2014).

Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G.M., Paxson, V. and Savage, S. (2008), "Spamalystic: an empirical analysis of spam marketing conversion", *15th ACM Conference on Computer and Communications Security*, Alexandria, VA

Lowe, W. (2006), *Orthopedic Assessment of Massage Therapy*, Daviau-Scott.

Philippovich, Y. (2013), "Computer related tools to keep communicative interaction between deaf people", *10th National Congress of the International Society of Applied Psycholinguistics*, Moscow.

Rafiqul, Z.K. and Noor, A.I. (2012), "Hand gesture recognition: a literature review", *International Journal of Artificial Intelligence & Applications*, Vol. 3 No. 4, pp. 161-174.

Sandler, W. and Lillo-Martin, D. (2001), "Natural sign languages", *Handbook of Linguistics*, Blackwell Publishers Ltd, Hong Kong, pp. 533-562.

Shumilov, A. and Philippovich, A. (2014), "An animated CAPTCHA that uses hand gestures", *Systemnii Administrator*, Vol. 4 No. 1, pp. 82-84.

Shumilov, A. and Philippovich, A. (2016), "Using 3D animated hand gestures to create a new type of CAPTCHA", *The 3rd International Conference on Analysis of Images, Social Networks and Texts (AIST' (2014)*, CEUR-WS.org, Yekaterinburg, pp. 151-155.

von Ahn, L. (2011), "Luis von Ahn: massive-scale online collaboration", available at: www.ted.com/talks/luis_von_ahn_massive_scale_online_collaboration (accessed 20 October 2014).

Vylomova, E., Philippovich, A., Danshina, M., Golubeva, I. and Philippovich, Y. (2014), "Neural models for recognition of basic units of semiographic chants", in Dmitry, M.Y. and Ignatov, I. (Eds), *Analysis of Images, Social Networks and Texts: 436 of Communications in Computer and Information Science*, Springer International Publishing, New York, NY, pp. 249-254.

Corresponding author
Artem Shumilov can be contacted at: ashumilov@it-claim.ru